Trojan-Resilience without Cryptography

Suvradip Chakraborty[2], Stefan Dziembowski[1], Małgorzata Gałązka[1], Tomasz Lizurej[1⋆], Krzysztof Pietrzak[2⋆⋆], Michelle Yeo[2]

[1] University of Warsaw
[2] IST Austria

Abstract. Digital hardware Trojans are integrated circuits whose implementation differ from the specification in an arbitrary and malicious way. For example, the circuit can differ from its specified input/output behavior after some fixed number of queries (known as "time bombs") or on some particular input (known as "cheat codes").
To detect such Trojans, countermeasures using multiparty computation (MPC) or verifiable computation (VC) have been proposed. On a high level, to realize a circuit with specification $\mathcal{F}$ one has more sophisticated circuits $\mathcal{F}^\diamond$ manufactured (where $\mathcal{F}^\diamond$ specifies a MPC or VC of $\mathcal{F}$), and then embeds these $\mathcal{F}^\diamond$'s into a *master circuit* which must be trusted but is relatively simple compared to $\mathcal{F}$. Those solutions impose a significant overhead as $\mathcal{F}^\diamond$ is much more complex than $\mathcal{F}$, also the master circuits are not exactly trivial.
In this work, we show that in restricted settings, where $\mathcal{F}$ has no evolving state and is queried on independent inputs, we can achieve a relaxed security notion using very simple constructions. In particular, we do not change the specification of the circuit at all (i.e., $\mathcal{F} = \mathcal{F}^\diamond$). Moreover the master circuit basically just queries a subset of its manufactured circuits and checks if they're all the same.
The security we achieve guarantees that, if the manufactured circuits are initially tested on up to T inputs, the master circuit will catch Trojans that try to deviate on significantly more than a $1/T$ fraction of the inputs. This bound is optimal for the type of construction considered, and we provably achieve it using a construction where 12 instantiations of $\mathcal{F}$ need to be embedded into the master. We also discuss an extremely simple construction with just 2 instantiations for which we conjecture that it already achieves the optimal bound.

1 Hardware Trojans

Preventing attacks on cryptographic hardware that are based on leakage and tampering has been a popular topic both in the theory in the practical research

⋆ Stefan Dziembowski, Małgorzata Gałązka, and Tomasz Lizurej were supported by the 2016/1/4 project carried out within the Team program of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

⋆⋆ Suvradip and Krzysztof have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815 - TOCNeT)

communities [11,10,9,8,16,7]. Despite being very powerful, the models considered in this area are restricted in the sense that it is typically assumed that a given device has been manufactured correctly, i.e., the adversary is present during the execution of the device, but not when it is produced. As it turns out, this assumption is not always justifiable, and in particular in some cases the adversary may be able to modify the device at the production time. This is because, for economic reasons, private companies and government agencies are often forced to use hardware that they did not produce themselves. The contemporary, highly-specialized digital technology requires components that are produced by many different enterprises, usually operating in different geographic locations. Even a single chip is often manufactured in a production cycle that involves different entities. In a very popular method of hardware production, called the *foundry model*, the product designer is only developing the abstract description of a device. The real hardware fabrication happens in *foundry*. Only few major companies (like Intel) still manufacture chips by themselves [17].

Modifications to the original circuit specification introduced during the manufacturing process (in a way that is hard to detect by inspection and simple testing) are called *hardware Trojans*, and can be viewed as the extreme version of hardware attacks. For more on the practical feasibility of such attacks the reader may consult, e.g., books [17,14], or popular-science articles [1,13]. Hardware Trojans can be loosely classified into *digital* and *physical* ones. Physical hardware Trojans can be triggered and/or communicate via a physical side-channel, while digital hardware Trojans only use the regular communication interfaces. In this paper we only consider digital hardware Trojans.

1.1 Detecting digital hardware Trojans

A simple non-cryptographic countermeasure to detect whether a circuit F contains a hardware Trojan or follows the specification F is testing: one samples inputs $x_1, \ldots, x_T$, queries $y_i \quad \mathsf{F}(x_i)$ and checks whether $y_i = F(x_i)$ for all i. Two types of digital hardware Trojans discussed in the literature that evade detection by such simple testing are time bombs and cheat codes (see, e.g., [6]). A *time bomb* is a hardware Trojan where the circuit starts deviating after a fixed number of queries. *Cheat codes* refer to hardware Trojans where the circuits deviate on a set of hard-coded inputs. To achieve some robustness against all digital hardware Trojans, solutions using cryptographic tools, in particular verifiable computation (VC) [18,2] and multiparty computation (MPC) [6] were suggested. In both cases the idea is to take the specification F of the desired circuit and replace it with a more sophisticated construction of one or more circuits F . The circuit(s) F that (presumably) are manufactured according to specification F are then embedded into a master circuit M to get a circuit M^{F} which is proven to follow specification F with high probability as long as it produces outputs. The master circuit must be trusted, but hopefully can be much simpler than F. We elaborate on these two methods below.

Method	**Specification of circuit $\mathcal{F}^\diamond$ (to be manufactured by untrusted fab) to realize $\mathcal{F}$**	**Test and (trusted) Master Circuit**	**Security with Trojans**	**Functionality when no Trojans present**
MPC [6]	**Different than $\mathcal{F}$**: Implements functionality $\mathcal{F}$ in terms of a 3-player MPC computation.	**Testing**: T **test queries.** **Master's computation — non-trivial**: Master secret shares inputs and reconstructs outputs. It needs lots of randomness.	Gives $n < T/2$ correct and guaranteed outputs with prob. $1 - \exp(-k)$. No restriction on $\mathcal{F}$.	**Limited**: Behaves like $\mathcal{F}$ for $n = T/2$ queries, then stops.
VC [18,2]	**Different than $\mathcal{F}$**: $\mathcal{F}$ plus succinct proof of correct computation	**Testing**: **no test queries.** **Master's computation — non-trivial**: Master verifies succinct proof.	No guarantee on number of outputs, but as long as outputs are provided they are correct. No restriction on $\mathcal{F}$	**Ideal**: behaves like $\mathcal{F}$ for unbounded number of outputs.
This work	**Same as $\mathcal{F}$**: 12 instantiations needed in our provable construction, 2 conjectured to be already sufficient.	**Testing**: T **test queries.** **Master's computation — very simple**: Master only does equality checks and Multiplexer. Needs tiny amount of randomness.	Either the Trojans will be detected with probability $1-o(1)$, or at most a $O(1/T)$ fraction of the outputs is wrong. Requires that $\mathcal{F}$ has no evolving state and inputs are independent.	**Ideal**: (as above)

Fig. 1. Comparison of cryptographic solutions with our new construction. We achieve weaker security, but with a *much* simpler construction.

Using verifiable computation. Here the idea is to let F (x) output a tuple $(y;$ $)$ where $y = F(x)$ and $$ is a succinct zero-knowledge proof (see [3]) that y is the correct output. In the compiled circuit M^{F} the master M in input x invokes $(y;$ $)$ F , then verifies the proof $$ and only outputs y if the check passes. If verification fails, the master aborts with a warning. As long as the compiled circuit provides outputs, they are guaranteed to be correct. If there are no Trojans, the number of outputs is unbounded; but if there is a Trojan, they can make the compiled circuit abort already at the first query. See [18,2] for the details.

Using multiparty computation. In this case, the idea is to use secure multiparty computation protocols (MPCs, see, e.g., [4]). The compiled circuit M^{F} contains some number of sub-components F that communicate only via the master circuit. In [6], this number is $3k$ (where k is a parameter). The sub-components are grouped in triples, each of them executing a 3-party protocol. In order to avoid the "cheat code" attacks, the master secret shares the input between the 3 parties. To get assurance that the sub-components are not misbehaving they are tested before deployment. In order to avoid the "time bomb" attacks, the number of times each sub-component is tested is an independently chosen random number from 1 to T. The output of each triple is secret-shared between its sub-components. Each of them sends its share to the master circuit, who reconstructs the k secrets, and outputs the value that is equal to the majority of these secrets. For the details see [6].

Simple schemes. In this work we consider compilers as discussed above, but only particularly simple ones which have the potential of being actually practical. In particular, we require that F F . That is, the specification F of the functionality given to the untrusted manufacturer is the actual functionality $F : X$ $!$ Y we want to implement. Moreover, our master just invokes (a random subset of) the circuits on the input and checks if the outputs are consistent.

This restricted model has very appealing properties. For example, it means one can use our countermeasures with circuits that have already been manufactured. But there are also limitations on what type of security one can achieve. Informally, the security we prove for our construction roughly states that for any constant $c > 0$ there exists a constant c' such that no malicious manufacturer can create Trojans which (1) will not be detected with probability at least c, and (2) if not detected, will output a c'/T fraction of wrong outputs. Here T is an upper bound on the number of test queries we can make to the Trojans before they are released.

In particular, we only guarantee that most outputs are correct, and we additionally require that the inputs are iid. Unfortunately, it's not hard to see that for the simple class of constructions considered these assumptions are not far from necessary.[3] We will state the security of the VC and MPC solutions using our notion of Trojan-resilience in §2.7.

[3] We show that a small fraction of wrong outputs must be allowed in §2.4. The iid assumption can be somewhat relaxed, but as we don't have a clean necessary condition we will not discuss this further in this paper. Informally, a sufficient condition

It's fair to ask whether our notion has any interesting applications at all. Two settings in which Trojan resilience might be required are (1) in settings where a computation is performed where false (or at least too many false) outputs would have serious consequences, and (2) cryptographic settings where the circuit holds a key or other secret values that should not leak.

For (1) our compiler would only be provably sufficient if the inputs are iid, and only useful if a small fraction of false outputs can be tolerated. This is certainly a major restriction, but as outlined above, if one doesn't have the luxury to manufacture circuits that are much more sophisticated than the required functionality, it's basically the best one can get. Depending on the setting, one can potentially use our compiler in some mode – exploiting redundancy or using repetition – to fix those issues. We sketch some measures in the cryptographic setting below.

For the cryptographic setting (2) our notion seems even less useful: if the adversary can learn outputs of the Trojans, he can use the $(1=T)$ fraction of wrong outputs to embed (and thus leak) its secrets. While using the compiler directly might not be a good idea, we see it as a first but major step towards simple and Trojan-resilient constructions in the cryptographic setting. As an example, consider a weak PRF $F : K \quad X \; ! \; Y$ (a weak PRF is defined like a regular PRF, but the outputs are only pseudorandom if queried on random inputs). While implementing $F(k;)$ using our compiler directly is not a good idea as discussed above,[4] we can compile $t > 1$ weak PRFs with independent keys and inputs and finally XOR the outputs of the t master circuits to implement a weak PRF $F_3((k_1;:::;k_t);(x_1;:::;x_t)) = \; _{i=1}^{t} F(k_i;x_i)$. Intuitively, the output can only leak significant information about the keys if *all* t outputs are wrong as otherwise the at least one pseudorandom output will mask everything. If each output is wrong with probability, say $1=T$ for a modest $T = 2^{30}$ and we use $t = 3$, then for each query we only have a probability of $1=T^3 = 2^{-90}$ that all $t = 3$ outputs deviate, which we can safely assume will never happen. Unfortunately, at this point we can't prove the above intuition and leave this for future work. For one thing, while we know that the XOR will not leak much if at least one of the t values is correct when the weak PRFs are modelled as ideal ciphers [12], we don't have a similar result in the computational setting. More importantly, we only prove that at most a $1=T$ fraction of the outputs is wrong once a sufficiently large number of queries was made, but to conclude that in the above construction all t instances fail at the same time with probability at most $1=T^t$ we need a stronger statement saying that for each individual query the probability of failure is $1=T$

seems to just require that there is no (efficiently recognisable) subset of inputs which appear rarely (not more than with probability around $1/T$) but can come in "bursts", say two such inputs are consecutive with prob. $\gg 1/T^2$.

[4] It's acceptable by our construction if the inputs are iid conditioned on some secret, so the master on input x and key k can forward (k, x) to the circuits. Alternatively the key k can be hard-coded in the circuit (probably not a good idea if the manufacturer is not trusted in the first place) or, if the circuits have some storage, one can give them k after receiving the circuits from the manufacturer.

(we believe that this is indeed true for our construction, but the current proof does not imply this).

Weak PRFs are sufficient for many basic symmetric-key cryptographic tasks like authentication or encryption[5]. Even if a fraction of outputs can be wrong, as long as they don't leak the key (as it seems to be the case for the construction just sketched), this will only affect completeness, but not security. An even more interesting construction, and the original motivation for this work, is a trojan-resilient stream cipher. This could then be used to e.g., generate the high amount of randomness required in side-channel countermeasures like masking schemes. The appealing property of a stream-cipher in this setting is that we don't care about correctness at all, we just want the output to be pseudorandom. It's not difficult to come up with a candidate for such a stream-cipher based on our compiler, but again, a proof will require more ideas. One such construction would start with the weak PRF construction just discussed, and then use two instantiations of it in the leakage-resilient mode from [15].

2 Definition and Security of Simple Schemes

For m 2 N, an *m-redundant simple construction* $_m = (\mathsf{T}\ ; \mathsf{M}\ )$ is specified by a master circuit $\mathsf{M}\ : X\ !\ Y\ [\ f\mathsf{abort}g$ and a test setup $\mathsf{T}\ : \mathsf{N}\ !\ f\mathsf{fail}; \mathsf{pass}g$.[6] The indicates that they expect access to some "oracles". The following oracles will be used: (a) $\mathsf{F}_1; :::; \mathsf{F}_m$ — the Trojan circuits that presumably implement the functionality $F : X\ !\ Y$, (b) F — a trusted implementation of F (only available in the test phase), and (c) $ — a source of random bits (sometimes we will provide the randomness as input instead),

2.1 Test and deployment

The construction $_m$ which implements F in a Trojan-resilient way using the untrusted $\mathsf{F}_1; :::; \mathsf{F}_m$ is tested and deployed as follows.

Lab Phase (test): In this first phase we execute $f\mathsf{pass}; \mathsf{fail}g$ $\mathsf{T}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\mathsf{F},\$}(T)$ The input T specifies that each F_i may be queried at most T times. If the output is fail, a Trojan was detected. Otherwise (i.e. the output is pass) we move to the next phase.

Wild Phase (deployment): If the test outputs pass, the F_i's are embedded into the master to get a circuit $\mathsf{M}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\$} : X\ !\ Y\ [$ abort:

2.2 Completeness

The completeness requirement states that if every F_i correctly implements F, then the test phase outputs pass with probability 1 and the master truthfully

[5] To encrypt m sample a random r and compute the ciphertext $(r, \mathcal{F}(k, r) \oplus m)$

[6] We consider much stronger $\mathsf{M}^*, \mathsf{T}^*$ for the lower bounds compared to what we require in the constructions as discussed in Sec. 2.5

implements the functionality F. That is, for every sequence $x_1, x_2, \ldots, x_q$ (of arbitrary length and potentially with repetitions) we have

$$\Pr[\forall i \in [q] : y_i = F(x_i)] = 1 \text{ where for } i = 1 \text{ to } q : y_i := \mathsf{M}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\$}(x_i)$$

The reason we define completeness this way and not simply for all x we have $\Pr[\mathsf{M}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\$}(x) = F(x)]$ is that the Trojan F_i can be stateful, so the order in which queries are made does matter.

2.3 Security of simple schemes

We consider a security game $\mathsf{TrojanGame}(\ ; T; Q)$ where, for some $T, Q \in \mathbb{Z}$, an adversary Adv can choose the functionality F and the Trojan circuits $\mathsf{F}_1, \ldots, \mathsf{F}_m$. We first run the test phase $\mathsf{T}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\mathsf{F},\$}(T)$ We then run the wild phase by quering the master on Q iid inputs $x_1, \ldots, x_Q$.

$$\text{for } i = 1, \ldots, Q : y_i \quad \mathsf{M}^{\mathsf{F}_1,\ldots,\mathsf{F}_m,\$}(x_i):$$

The goal of the adversary is two-fold:

1. They do not want to be caught, if either $= \mathsf{fail}$ or $y_i = \mathsf{abort}$ for some $i \in [Q]$ we say the adversary was detected and define the predicate
$$\mathsf{detect} = \mathsf{false} \quad ( ) \quad (\quad = \mathsf{pass}) \wedge (\forall i \in [Q] : y_i \neq \mathsf{abort})$$
2. They want the master to output as many wrong outputs as possible. We denote the number of wrong outputs by $Y \stackrel{\text{def}}{=} |\{i : y_i \neq F(x_i)\}|$.

Informally, we call a compiler (like our simple schemes) $(\mathsf{win}, \mathsf{wrng})$-Trojan resilient, or simply $(\mathsf{win}, \mathsf{wrng})$-secure, if for every Trojan, the probability that it causes the master to output wrng fraction of wrong outputs without being detected is at most win. In the formal definition win and wrng are allowed to be a function of the number of test queries T.

Definition 1 ($(\mathsf{win}, \mathsf{wrng})$-Trojan resilience). *And adversary* $(\mathsf{win}, \mathsf{wrng})$*-wins in* $\mathsf{TrojanGame}(\ ; T; Q)$ *if the master outputs more than a* wrng *fraction of wrong values without the Trojans being detected with probability greater than* win*, i.e.,*

$$\Pr_{\mathsf{TrojanGame}(\Pi,T,Q)}[(\mathsf{detect} = \mathsf{false}) \wedge (Y/Q \quad \mathsf{wrng})] \quad \mathsf{win}$$

For $\mathsf{win} : \mathbb{N} \to [0,1]$; $\mathsf{wrng} : \mathbb{N} \to [0,1]$; $q : \mathbb{N} \to \mathbb{N}$, *we say that* is $(\mathsf{win}(T), \mathsf{wrng}(T), q(T))$-Trojan-resilient (or simply "secure") *if there exists a constant* T_0*, such that for all* $T \quad T_0$ *and* $Q \quad q(T)$ *no adversary* $(\mathsf{win}(T), \mathsf{wrng}(T))$*-wins in* $\mathsf{TrojanGame}(\ ; T; Q)$.

We say is $(\mathsf{win}(T), \mathsf{wrng}(T))$ *Trojan-resilient if it is* $(\mathsf{win}(T), \mathsf{wrng}(T), q(T))$*-Trojan-resilient for some (su ciently large) polynomial* $q(T) \in \mathsf{poly}(T)$.

In all our simple constructions the test and master only use the outputs of the F_i (and for the test also F) oracles to check for equivalence. This fact will allow us to consider somewhat restricted adversaries in the security proof.

Definition 2 (Generic Simple Scheme). *A* generic *simple scheme* $\mathsf{T}\ ;\mathsf{M}$ *treats the outputs of the* F_i *(and for* T *additionally* F*) oracles like variables. Concretely, two or more oracles can be queried on the same input, and then one checks if the outputs are identical. Moreover the master can use the output of an* F_i *as its own output.*

By the following lemma, to prove security of generic simple schemes, it will be sufficient to consider restricted adversaries that always choose to attack the trivial functionality $F(x) = 0$ and where the output range of the Trojans is a bit.

Lemma 1. *For any generic simple scheme* $\ _m$*, assume an adversary* Adv *exists that* $(\mathsf{win};\mathsf{wrng})$*-wins in* $\mathsf{TrojanGame}(\ _m; T; Q)$ *and let* $F : X \ ! \ Y\ ;\ \mathsf{F}_1;:::;\mathsf{F}_m : X \ ! \ Y$ *denote its choices for the attack. Then there exists an adversary* Adv^0 *who also* $(\mathsf{win};\mathsf{wrng})$*-wins in* $\mathsf{TrojanGame}(\ _m; T; Q)$ *and chooses* $F^0 : X \ ! \ f0;1g\ ;\ \mathsf{F}_1^0;:::;\mathsf{F}_m^0 : X \ ! \ f0;1g$ *where moreover* $8x \ 2 \ X : F^0(x) = 0$*.*

Proof. Adv^0 firstly runs Adv to learn *(i)* the functionality $F : X \ ! \ Y$ which it wants to attack and *(ii)* its Trojans $\mathsf{F}_1;:::;\mathsf{F}_m$. It then outputs (as its choice of function to attack) an F^0 where $8x \ 2 \ X : F^0(x) = 0$ and, for every $i \ 2 \ [m]$, it chooses the Trojan F_i^0 to output 0 if F_i would output the correct value, and 1 otherwise. More formally, $\mathsf{F}_i^0(x)$ invokes the original Trojan $y \quad \mathsf{F}_i(x)$ and outputs 0 if $F(x) = y$ and 1 otherwise.

By construction, whenever one of the F_i^0's deviates (i.e., outputs 1), also the original F_i would have deviated. And whenever the test or master detect an inconsistency in the new construction, they would also have detected an inconsistency with the original F and F_i.[7]

2.4 Lower bounds

By definition, $(\mathsf{win};\mathsf{wrng})$-security implies $(\mathsf{win}^0;\mathsf{wrng}^0)$-security for any $\mathsf{win}^0 \quad \mathsf{win};\mathsf{wrng}^0 \quad \mathsf{wrng}$. The completeness property implies that no scheme is $(1;0)$-secure (as by behaving honestly an adversary can $(1;0)$-win). And also no scheme is $(0;1)$-secure (as $\Pr[E] \quad 0$ holds for every event E). Thus our $(\mathsf{win};\mathsf{wrng})$-security notion is only interesting if both, win and wrng are > 0. We will prove the following lower bound:

Lemma 2 (Lower bound for simple schemes). *For any* $c > 0$ *and* $m \ 2$ N *there exists a constant* $c^0 = c^0(c;m) > 0$ *such that no* m*-redundant simple scheme* $\ _m$ *is* $(c;\frac{c^0}{T})$*-Trojan-resilient.*

Proof. Adv chooses the constant functionality $F(x) = 0$ with a sufficiently large input domain $jXj \quad (m \ T)^2$ (so that sampling $m \ T$ elements at random from

[7] Let us mention that the opposite is not true (it's possible that for some $i \neq j$ we have $\mathsf{F}'_i(x) = \mathsf{F}'_j(x) = 1$, while $\mathsf{F}_i(x) \neq \mathsf{F}_j(x)$). This just captures the observation that an adversary who wants to deviate as often as possible without being detected can wlog. always deviate to the same value.

X with or without repetition is basically the same). Now Adv samples a random subset $X' \subset X$, $|X'| = |X| \cdot \frac{1.1\, c'}{T}$ (for c' to be determined) and then defines Trojans which deviate on inputs from X'

$$\forall i \in [m] : F_i(x) = \begin{cases} 1 \text{ if } x \in X' \text{ (deviate)} \\ 0 \text{ if } x \notin X' \text{ (correct)} \end{cases}$$

Should the test pass, the master will deviate on each input with probability $1.1\ c'/T$, if we set the number of queries Q large enough, the fraction of wrong outputs will be close to its expectation $1.1\ c'/T$, and thus almost certainly larger than c'/T.

It remains to prove that the test passes with probability $\geq c$. By correctness, the testing procedure $T^{F_1,\ldots,F_m,F,\$}$ must output *pass* unless one of the total $m \cdot T$ queries it made to the F_is falls into the random subset X'. The probability that no such query is made is at least

$$1 - \frac{1.1\ c'}{T}^{m \cdot T}$$

and this expression goes to 1 as c' goes to 0. We now choose $c' > 0$ sufficiently small so the expression becomes $\geq c$. To get a quantitive bound one can use the well known inequality $\lim_{T \to \infty} (1 - 1/T)^T = 1/e \approx 0.367879$

The (proof of) the previous lemma also implies the following.

Corollary 1. If a simple scheme $_m$ is $(\mathrm{win}(T), \mathrm{wrng}(T))$ secure with

1. $\mathrm{win}(T) \geq 1 - o(1)$ then $\mathrm{wrng}(T) \geq o(1/T)$.
2. $\mathrm{wrng}(T) \geq \omega(1/T)$ then $\mathrm{win}(T) \geq o(1)$.

The first item means that if Adv wants to make sure the Trojan is only detected with sub-constant probability, then he can only force the master to output a $o(1/T)$ fraction of wrong outputs during deployment. The second item means that if Adv wants to deviate on a asymptotically larger than $1/T$ fraction of outputs, it will be detected with a probability going to 1.

Not interesting security for 1-redundant schemes. For $m = 1$ redundant circuits a much stronger lower bound compared to Lemma 2 holds. The following Lemma implies that no 1-redundant scheme is $(\varepsilon(T), \delta(T))$-Trojan-resilient for any $\varepsilon(T) > 0$ and $\delta(T) = 1/\mathrm{poly}(T)$ (say $\varepsilon(T) = 2^{-T}$; $\delta(T) = T^{-100}$).

Lemma 3 (Lower bound for $m = 1$). For any 1-redundant scheme $_1$ and any polynomial $p(T) > 0$, there is an adversary that $(1, 1 - 1/p(T))$-wins in the $\mathrm{TrojanGame}(_1, T, Q)$ game for $Q \leq p(T) \cdot T$.

Proof. Consider an adversary who chooses a time bomb Trojan F_1 which correctly outputs $F(x)$ for the first T queries and also stores those queries, so it can output the correct value if one of those queries is repeated in the future. From query $T + 1$ the Trojan outputs wrong values unless it is given one of the first

T queries as input, in which case it outputs the correct value. This Trojan will pass any test making at most T invocations, while the master will deviate on almost all queries, i.e., all except the first T.

To see why we store the first T queries and do not deviate on them when they repeat in the future, consider a master which stores the outputs it observes on the first T queries so it can later detect inconsistencies.

2.5 Efficiency of lower bound vs. constructions

For the lower bounds in the previous section, the only restriction on the test $T^{F_1,\ldots,F_m,F,\$}(T)$ is that each F_i can only be queried at most T times. There are no restrictions on the master $M^{F_1,\ldots,F_m,\$}()$ at all. In particular, it can be stateful, computationally unbounded, use an arbitrary amount of randomness, and query the F_is on an unbounded number of inputs (as the Trojan F_is can be stateful this is not the same as learning the function table of the F_i's).

While the lack of any restrictions makes the lower bound stronger, we want our upper bounds, i.e., the actual constructions, to be as efficient (in terms of computational, query and randomness complexity) and simple as possible, and they will indeed be very simple.

Let us stress that one thing the definition does not allow is the test to pass a message to the master. If we would allow a message of unbounded length to be passed this way no non-trivial lower bound would hold as T could send the entire function table of F to M, which then could perfectly implement F. Of course such a construction would get against the entire motivation for simple schemes where M should be much simpler and independent of F. Still, constructions where the test phase sends a short message to the master (say, a few correct input/output pairs of F which the master could later use to audit the Trojans) could be an interesting relaxation to be considered.

2.6 Our results and conjectures

Our main technical result is a construction of a simple scheme which basically matches the lower bound from Lem. 2. Of course for any constant $c > 0$, the constant c' in the theorem below must be larger than in Lem. 2 so there's no contradiction.

Theorem 1 (Main, optimal security of $_{12}$). For any constant $c > 0$ there is a constant c' such that the simple construction $_{12}$ from Fig. 3 is $(c; \frac{c'}{T})$-Trojan resilient.

While $(c; \frac{c}{T})$-Trojan-resilience matches our lower bound, the construction is $m = 12$-redundant (recall this means we need 12 instantiations of F manufactured to instantiate the scheme). While for $m = 1$ redundancy is not sufficient to get any interesting security, as we showed in Lem. 3, we conjecture that $m = 2$ is sufficient to match the lower bound, and give a candidate construction.

Conjecture 1 (Optimal security of $_2$). For any $0 < < 1$ and any constant $c > 0$ there is a constant $c' = c'(c;)$ such that the simple construction $_2$ from Sec. 3 is $(c; \frac{c'}{T})$-Trojan resilient.

Fig. 2. Construction $\overset{?}{2}$ (discussed in Sec. 2.9), which is $(c; \frac{c^0}{T})$ secure for history-independent Trojans.

Fig. 3. Construction $_{12}$ for which we prove optimal Trojan-resilience as stated in Thm. 1. Very informally, the security proof is by contradiction: via a sequence of hybrids an attack against $_{12}$ is shown to imply an attack where the yellow part basically corresponds to $\overset{?}{2}$ with two history independent circuits. This attack contradicts the security of $\overset{?}{2}$ as stated in Thm. 2.

The parameter in this construction basically specifies that the master will query both oracles F_1 and F_2 on a (random) T fraction of the input, and check consistency in this case. While the conjecture is wrong for $= 0$ and 1, the $= 0$ case (i.e., when we always query both F_1 and F_2) will be of interest to us as security of the $\stackrel{?}{2} \stackrel{def}{=} \stackrel{0}{2}$ construction against a limited adversary (termed history-independent and discussed in Sec. 2.9 below) will be a crucial step towards our proof of our main theorem.

2.7 Comparison with VC and MPC

Let us shortly compare the security we achieve with the more costly solutions based on verifiable computation (VC) [18,2] and multiparty computation (MPC) [6] discussed in the introduction. We can consider $(\text{win}; \text{wrng})$-security as in Definition 1 also for the VC and MPC construction, here one would need change the $\text{TrojanGame}(\ ; T; Q\ )$ from Ÿ2.3 to allow the trojans F_i to implement a different functionality than the target F (for VC one needs to compute an extra succinct proof, for MPC the trojans implement the players in an MPC computation). For VC there's no test (so $T = 0$) and only one $m = 1$ Trojan, and for MPC and VC we can drop the requirement that the inputs are iid.

In the VC construction the master will catch every wrong output (except with negligible probability), so for any polynomial poly there is a negligible function negl (in the security parameter of the underlying succinct proof system), such that the scheme is $(1/\text{poly}; \text{negl}; Q)$ secure for any polynomial Q.

For the MPC construction the master will provide $Q < c_0 T$ outputs with probability c_1^m (where $c_0 \in [0; 1/2]$ and $c_1 \in [0; 1]$ are some constants), but while outputs are provided they are most likely correct, so for any polynomial $Q; T$ we have $(1 \quad c_1^m; \text{negl}; Q)$ security.

2.8 Stateless Trojans

In our security definition we put no restriction on the Trojans F_i provided by the adversary (other than being digital hardware Trojans as discussed in the introduction), in particular, the F_i's can have arbitrary complex evolving state while honestly manufactured circuits could be stateless. We can consider a variant of our security definition (Def. 1) where the adversary is only allowed to choose stateless Trojan circuits F_i. Note that the lower bound from Lem. 2 still holds as in its proof we did only consider stateless F_i's. There's an extremely simple 1-redundant construction that matches the lower bound when the adversary is only allowed to chose stateless Trojans.

Consider a construction $_1 = (T\ ; M\ )$ where the master is the simplest imaginable: it just forwards inputs/outputs to/from its oracle, if F_1 is stateless this simply means $M^{F_1}(\ ) = F_1(\ )$. The test $T^{F_1;F;\$}(T)$ queries F_1 and the trusted F on T random inputs and outputs fail if there is a mismatch.

Proposition 1 (Optimal security for 1-redundant scheme for stateless Trojans). For any constant $c > 0$ there is a constant $c' > 0$ such that $_1$

is $(c, \frac{c'}{T})$-Trojan resilient if the adversary is additionally restricted to choose a stateless Trojan.

Proof. If $wrng'$ denotes the fraction of inputs on which the Trojan F_1 differs from the specification F (both chosen by an adversary Adv, note that $wrng'$ is only well defined here as F_1 is stateless), then $wrng'$ must satisfy $c > (1 - wrng')^T$ if the adversary wants to $(c, wrng)$-win for any $wrng$, as otherwise already the test catches the Trojan with probability $(1 - wrng')^T > c$. For $c > (1 - wrng')^T$ to hold $wrng' \geq (1 - c)/T$, in particular, $wrng' \geq c'/T$ for some $c' > 0$.

2.9 History-independent Trojans

A notion of in-between general (stateful) Trojans and stateless Trojans will play a central role in our security proof. We say a trojan F_i is history-independent if its only state is a counter which is incremented by one on every invocation, so it's answer to the i'th query can depend on the current index i, but not on any inputs it saw in the past.

We observe that Lem. 3 stating that no 1-redundant simple scheme can be secure still holds if we restrict the choice of the adversary to history-independent Trojans as the time-bomb Trojan used in the proof is history-independent. We will show a 2-redundant construction $\Pi_2^?$ that achieves optimal security against history-independent Trojans.

Theorem 2 (History-Independent Security of $\Pi_2^?$). For any constant $c > 0$ there is a constant $c' = c'(c) > 0$ such that $\Pi_2^?$ from Fig. 2 is $(c, \frac{c'}{T})$-Trojan resilient if the adversary is additionally restricted to choose a history-independent Trojans.

The technical Lem. 4 we prove and which implies this theorem, actually implies a stronger statement: for any positive integer k, the above holds even if we relax the security notion and declare the adversary a winner as long a Trojan is detected by the test or master at most $k - 1$ times. What this exactly means is explained in Sec. 4.2.1. Note that this notion coincides with the standard notion for $k = 1$.

The $\Pi_2^?$ scheme is just the Π_2 scheme from Conjecture 1 for $= 0$, where we conjecture that Π_2 is (in some sense) optimally secure for $0 < < 1$. For the conjecture is wrong, but somewhat ironically we are only able to prove security against history-independent Trojans for $= 0$, and this result will be key towards proving the security of Π_{12} as stated in our main Thm. 1.

2.10 Proof outline

The proof of our main Thm. 1 stating that Π_{12} is optimally Trojan-resilient is done in two steps. As just discussed, we first prove security of $\Pi_2^?$ against history-independent Trojans, and in a second step we reduce the security of Π_{12} against general Trojans to the security of $\Pi_2^?$ against history-independent Trojans. We outline the main ideas of the two parts below.

Part 1: security of $\Pi_2^?$ against history-independent Trojans (Thm. 2,Lem. 4). $\Pi_2^?$ is a very simple scheme where the test $T^{F_1,F_2,F,\$}$ just queries F_1 on a random number $; \ 0 < T$ of inputs and checks if they are correct (the test ignores F_2). The master $M^{F_1,F_2,\$}(x)$ queries $y \ F_1(x)$ and $y' \ F_2(x)$ on x and aborts if they disagree.

In the proof of Lem. 4 we de ne p_i and q_i as the probability that F_1 and F_2 outputs a wrong value in the ith query on a random input, respectively. As $F_1; F_2$ are history independent, this is well de ned as this probability only depends on i (but not previous queries).

Let the variable denote the number of times the Trojans will be detected conditioned on the random number of test queries being . This value is (below Q is the number of queries to the master and we use the convention $q_i = 0$ for $i < 0$)

$$E[\ ] = \sum_{i=1}^{Q+} j p_i \quad q_i \quad j \qquad (1)$$

In this sum, the rst terms account for the test, and the last Q terms for the wild-phase. Moreover let Y denote the number of times F_1 deviates (and thus the master outputs a wrong value), its expectation is

$$E[Y \ ] = \sum_{i= \ +1}^{Q+} p_i$$

To prove Trojan-resilience of $\Pi_2^?$ as stated in Lem. 4 boils down to proving that, for most , whenever the probability of $= 0$ (i.e., the Trojan is not detected) is constant, the fraction of wrong outputs $Y =Q$ must be small (concretely, $O(1=T)$).

The core technical result establishing this fact is Lem. 5. Unfortunately, this Lem. only establishes this fact for the expectation, i.e., whenever $E[\ ]$ is small, also $E[Y \ ]$ is small. Here is where we use the fact that the $F_1; F_2$ are history independent: in the history independent case and Y can be written as the sum of independent boolean variables, so using a Cherno bound it follows that their actual value will be close to their expectation with high probability.

It is instructive to see why for example setting $p_i = q_i =$ for some xed > 0 does not contradict Thm. 2. To contradict it, the fraction of wrong outputs (which here is simply) must be $! (1=T)$. In this case, $E[\ ] = = ! (=T)$, which to contradict the lemma must be at least constant, which in turn means $2 o(T)$ must hold. As $; \ 0 < T$ is uniform, t's $o(T)$ with $o(1)$ probability, but for a contradiction we also need this probability to be constant.

Part 2: reducing the security of Π_{12} against general Trojans to the security of $\Pi_2^?$ against history independent Trojans (Thm. 1). While the random shift makes $\Pi_2^?$ secure against history-independent attacks (like time-bombs, where a Trojan starts deviating after some xed number of queries), it succumbs to cheat codes: as the master always queries $F_1; F_2$ on the same inputs, a Trojan can

specify some set of trigger inputs, and after receiving such a trigger the Trojans will deviate forever. By making the fraction of inputs that are triggers sparse, the Trojans will likely not be detected during testing (a $1/T$ fraction will survive testing with constant probability).

To prevent such a coordination via the inputs, in $_{12}$ inputs are somewhat randomly assigned to the di erent Trojans. In particular, as emphasized in the yellow area in Fig. 3, the F_1 is always queried on the input x, and then the random bit b determines whether F_2 or F_3 are queried on x. If an input x were to trigger the Trojans to always deviate, F_1 and one of F_2 and F_3 will be triggered, say it's F_2. But now, as soon as F_3 is queried in a future round the master will abort as F_1 will deviate, but F_3 will not (except if this query also happens to be a trigger, which is unlikely as triggers must be sparse to survive the testing phase).

This just shows why a particular attack does not work on $_{12}$. But we want a proof showing security against all possible Trojans. Our proof proceeds by a sequence of hybrids, where we start with assuming a successful attack on $_{12}$, and then, by carefully switching some circuits and rede ning them by hard coding "fake histories", we arrive at a hybrid game where there is still a successful attack, but now the circuits in the yellow area basically correspond to two the $^{?}_{2}$ construction instantiated with history-independent Trojans, but such an attack contradicts our security proof for $^{?}_{2}$ as stated in Lem. 4.

3 Conjectured Security of 2-redundant Schemes

While the main technical result in this paper is a simple scheme $_{12}$ that provably achieves optimal security as stated in Thm. 1, this construction is not really practical as it is 12-redundant. Recall that k-redundant means the master needs k instantiations of the functionality F, so it's in some sense the hardware cost. For this section let us also de ne a computational cost: the "rate" of a simple construction is the average number of invocations to its F_i oracles the master $M^{F_1,\ldots,F_m,\$}(\ )$ makes with any query.

3.1 A 2-redundant scheme $_2$

We will now de ne a scheme $_2$ which in terms of redundancy and rate is as e cient as we possibly could hope for a scheme with non-trivial security: it's 2-redundant and has a rate of slightly above (the trivial lower bound of) 1. The construction $_2 = (M, T)$, where $\ 2\ R;\ \ 0$ is illustrated in Fig. 4.

test: In the test phase, $T^{F_1,F_2,F}(T)$ picks a random $;\ 0\ \ T\ \ 1$, then queries F_1 on random inputs and checks if the outputs are correct by comparing with the trusted F.

master: With probability $1\ \ T$ the master $M^{F_1,F_2,\$}(x)$ picks either F_1 or F_2 at random, queries it on x and uses the output as its output. Otherwise, with probability T, the master queries both oracles and outputs abort if their outputs don't match, and forwards the output of F_1 otherwise.

Fig. 4. Construction $_2$ from Conjecture 1.

Our Conjecture 1 states that this construction achieves optimal security (optimal in the sense of matching the lower bound from Lem. 2) for any $0 < < 1$, i.e.,

> For any $0 < < 1$ and any constant $c > 0$ there is a constant c^0 such that $_2$ is $(c; \frac{c^0}{T})$-Trojan resilient.

We discuss how $_2$ performs against typical Trojans like time-bombs and cheat codes. Our conjecture only talks about $(\text{win}(T); \text{wrng}(T))$-security where the winning probability $\text{win}(T) = c$ is a constant, and here the exact value of does not seem to matter much as long as it is bounded away from 0 and 1. For $\text{win}(T) = o(1)$ the parameter will matter as those attacks will illustrate. (the $o(1)$ always denotes any value that goes to 0 as $T \to 1$).

Proposition 2 (Time Bomb against $_2$). For any , there exists an adversary that $((T); 1 o(1))$-wins in TrojanGame$((_2); T; ! (T))$

Proof (sketch). Let Adv choose the constant functionality $8x \ 2 \ X : F(x) = 0$, and a Trojan F_1 which outputs the correct value 0 for the rst T queries, and 1 for all queries $> T$, while F_2 always outputs 1.

F_1 will always pass the test. The master will abort i one of its rst T queries to F_1 (where the output is 0) is a $b = 2$ query (as then $F_1(x) = 1 \ 6 = 0 = F_2(x)$). With probability T we have $T \ T$, and in this case such a bad event only happens with constant probability (using $(1)^{1=} 1=e = 0:368:::$). So the Trojan will not be detected with probability $T =e$, and in this case also almost all outputs will be wrong.

Proposition 3 (Cheat Code against $_2$). For any , there exists an adversary that $((T ^{1}); 1 o(1))$-wins in TrojanGame$((_2); T; ! (T))$

Proof (sketch). Let Adv choose the constant functionality $\forall x \in X : F(x) = 0$. The Trojans F_1, F_2 output 0 until they get a query from a trigger set $X' \subset X$, after this query they always deviate and output 1.

If we set $|X'|/|X| = 1/T$, then the test will pass with constant probability $(1 - 1/T) \approx (1 - 1/T)^T \approx 1/e$. Assuming the Trojans passed the test phase, the master will not catch the Trojans if the first trigger query to F_1 and F_2 happen in-between the same $b = 2$ queries (or in such a query). This happens with probability $\approx T^{-1}$.

The two propositions above imply that the adversary can always $(T^{\max\{\cdot, \cdot - 1\}}, 1 - o(1))$-win by either using a time-bomb or cheat-code depending on $\cdot$. The winning probability is minimized if $\cdot = \cdot - 1$ which holds for $\cdot = 0.5$. We conjecture that the above two attacks are basically all one can do to attack $\Pi_2^?$.

Conjecture 2 (Security of $\Pi_2^{0.5}$ for low winning probabilities). For $\mathrm{win}(T) \geq 1/T^{0.5}$, $\Pi_2^{0.5}$ is $(\mathrm{win}(T), \mathrm{wrng}(T))$-Trojan resilient for some $\mathrm{wrng}(T) \in o(1)$.

4 A Scheme for History-Independent Trojans

In this section we define the simple scheme $\Pi_2^?$ and prove its security as claimed in Thm. 2. Recall that a history-independent Trojan circuit is a stateless circuit, except that it maintains a counter. We recall that a trojan is history-independent if its state is a counter which is incremented by one on every invocation, so its answer to the i'th query can depend on the current index i and current input x_i, but not on any inputs it saw in the past.

4.1 Notation

For an integer n we define $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$ and $[n]_0 \stackrel{\text{def}}{=} \{0, \ldots, n\}$. We will also use the Chernoff bound that for completeness appears in Appx. B.

4.2 Security of $\Pi_2^?$

4.2.1 Relaxing the winning condition. We can think of the security experiment $\mathrm{TrojanGame}(\Pi_2^?, T, Q)$ as proceeding in rounds. First, for a random $\cdot \in [T]$, we run the test for $\cdot$ rounds (in each quering F_1 and F on a random input and checking equivalence), and then Q rounds for quering the master (in each round quering F_1 and F_2 and checking for equivalence). The adversary immediately loses the game if a comparison fails (outputs 0 in the test or abort in the master) in any round.

We consider a relaxed notion of $(\mathrm{win}, \mathrm{wrng})$-winning, relaxed as we make it easier for the adversary, and thus proving security against this adversary gives a stronger statement. We define $(\mathrm{win}, \mathrm{wrng})$-$k$-winning like $(\mathrm{win}, \mathrm{wrng})$-winning, but the adversary is allowed to be detected in up to $k - 1$ rounds, so $(\mathrm{win}, \mathrm{wrng})$-1-winning is $(\mathrm{win}, \mathrm{wrng})$-winning.

This relaxed notion is not of practical interested, as one would immediately abort the moment a Trojan is detected. We consider this notion as we need it for the security proof of our main Thm. 1, where we will only be able to reduce security of $_{12}$ to the security of $^{?}_{2}$ (against history-independent Trojans) if $^{?}_{2}$ satis es this stronger notion.

4.2.2 Proof of Thm. 2. The following lemma implies Thm. 2 for k = 1, as discussed after the statement of the theorem the lemma below is somewhat more general as we'll need the stronger security for any k.

Lemma 4. For any constant $c > 0$ and positive integer k, there exists a constant c^0, and integer T_0 and polynomial $q(:)$ such that no adversary Adv exists that only chooses history-independent Trojans and that for any

$$T \quad T_0 \quad ; \quad Q \quad q(T)$$

can $(c; c^0=T)$-k-win TrojanGame$(\ ^{?}_{2}; T; Q)$.

Proof. For a given $c > 0$ de ne

$$c^{00} = \max f\, 64k; \quad 256 \ln(c=2)g$$

we then set $c^0; q(T)$ and T_0 as

$$c^0 = c^{00}=c^2 \ ; \ q(T) = \frac{5 \quad T^2 c}{c^{00}} + 5T \ ; \ T_0 = 1 \qquad (2)$$

These values are just chosen so that later our inequalities work out nicely, we did not try to optimise them.

By Lem. 1 we can consider the security experiment where an adversary Adv chooses the constant functionality $F : X \ ! \ 0$ as target and the two (history-independent) Trojans $F_1; F_2 : X \ ! \ f \ 0; 1g$ output a bit (so they can either correctly output 0 or deviate by outputting 1). As the $F_1; F_2$ are history independent, we can think of F_1 as a sequence $F^1_1; F^2_1; : : :$ of functions where F^i_1 behaves like F_1 on the ith query. Let p_i and q_i denote the probability that F_1 and F_2 deviates on the ith query, respectively

$$p_i \overset{def}{=} \Pr_{x \ X}[F^i_1(x) = 1] \quad ; \quad q_i \overset{def}{=} \Pr_{x \ X}[F^i_2(x) = 1]$$

In TrojanGame$(\ ^{?}_{2}; T; Q)$, for 2 $[T \quad 1]_0$ let the variable Y denote the number of wrong outputs by F_1 conditioned on the number of test queries $_{\$}$ $[T \quad 1]_0$ being = . The expectation is

$$E(Y) = \sum_{i=1}^{Q} p_{i+} \qquad (3)$$

Let the variables ^{T} and ^{M} denote the number of times the test and the master catch a Trojan conditioned on =

$$E[\ ^{T}] = \sum_{i=1} jp_i j \quad ; \quad E[\ ^{M}] \sum_{i=1}^{Q} jp_{i+} \quad q_i j$$

let $\stackrel{\text{def}}{=} T + M$ denote the total number of times the Trojans are detected, and ^{0} being the same but we ignore the last queries. With the convention that $q_i = 0$ for $i < 1$

$$E[\ ] \quad \sum_{i=1}^{Q+} jp_i \quad q_i \quad j \ ; \quad E[\ ^0] \quad \sum_{i=1}^{Q} jp_i \quad q_i \quad j \ ; \quad E[\ ] \quad E[\ ^0] \tag{4}$$

As we considerhistory-independent Trojans the $Y\ ;$ variables are sums of independent Bernoulli random variables. Using a Cherno bound we will later be able to use the fact that for such variables are close to their expectation with high probability.

Claim. For any $2 [T \quad 1]_0; \quad 2 [T \quad 1 \quad]$ (so $+ \quad T \quad 1$)

$$E[\ ] + E[\ _+\ ] \quad \frac{E[Y\ ] \quad T}{Q + T} \tag{5}$$

Proof (of Claim). Assume for a moment that $p_1; : : : ; p \ = 0$ as required to apply Lem. 16, then

$$E[\ ] + E[\ _+\ ] \overset{(4)}{\quad} E[\ ^0] + E[\ ^0_+\ ] \tag{6}$$

$$\sum \quad \sum_{2f\ ;\ +\ g\ i=1}^{Q} jp_i \quad q_i \quad j \tag{7}$$

$$\overset{\text{Lem: 5}}{\quad} \frac{\sum_{i=1}^{Q} p_i}{Q} = \quad \frac{E[Y_0]}{Q} \tag{8}$$

$$\frac{E[Y\ ] \quad T}{Q} \tag{9}$$

The last step used $E[Y_0] + \quad E[Y\ ]$ and $\quad T$.

We now justify our assumption $p_1; : : : ; p \ = 0$. For this change the security experiment and replace the Trojans $F_1; F_2$ chosen by the adversary with Trojans that rst behave correctly for the rst T inputs, and only then start behaving like $F_1; F_2$ (technically, the new Trojans deviate with probabilities $p_i^0; q_i^0$ satisfying $p_1^0; : : : ; p_T^0 = 0 ; q_1^0; : : : ; q_T^0 = 0$ and for $i > T \quad : : p_i^0 = p_{i \quad T}$ and $q_i^0 = q_{i \quad T}$). At the same time, we increase Q to $Q + T$. This change leaves $E[Y\ ]$ una ected, while $E[\ ^0]; E[\ ^0_+\ ]$ can only increase. This proves the claim, note that in (5) the denominator is $Q + T$ not Q as in (9) to account for this shift. 4

Claim. For all but at most a $c=2$ fraction of the $2 [T]_0$

$$E[\ ] \quad \frac{c \quad T}{8} \quad \frac{E[Y\ ] \quad T}{Q + T} \tag{10}$$

Proof (of Claim). We use Eq. (5) which can be understood as stating that if $E[\ ]$ is small for some , then all $E[\ _0]$ with $j \quad ^0 j$ large enough can't be

small too. Concretely, consider any for which (if no such exists the claim already follows)

$$E[\quad] < \frac{c\ T}{8} \quad \frac{E[Y\ ]\quad T}{Q + T}$$

then for all $^{0}2\ [T]_0$ for with $j\quad {}^{0}j \quad \frac{c\ T}{4}$ (note this is at least a c=2 fraction) by Eq. (5)

$$E[\quad] + E[\quad {}_{0}] \quad \frac{c\ T}{4} \quad \frac{E[Y\ ]\quad T}{Q + T}$$

the two equations above now give

$$E[\quad {}_{0}] \quad \frac{c\ T}{8} \quad \frac{E[Y\ ]\quad T}{Q + T}$$

as claimed. 4

To prove the lemma we need to show that when Q is su ciently large, any adversary attacking at least $c^{0}=T$ fraction of times, can win at most k times with probability less than c. Since the duration of test phase is chosen randomly from the set $f\,0; :::; T \quad 1g$, we start with the following equation:

$$\frac{1}{T}\sum_{=0}^{T\ 1} Pr[(Y = Q \quad c^{0}=T) \wedge (\quad < k)] < c \qquad (11)$$

Let c denote the probability the adversary k-wins conditioned on =

$$c \stackrel{def}{=} Pr[(Y = Q \quad c^{0}=T) \wedge (\quad < k)] \qquad (12)$$

With this notation we need to show

$$\frac{1}{T}\sum_{=0}^{T\ 1} c\ < c$$

which follow from the claim below

Claim. $c\ < c=2$ holds for all , except (the at most c=2 fraction of) the $2\ [T]_0$ for which (10) does not hold

Proof (of Claim). Consider any for which (10) holds. If for this $Pr[Y \quad Q\ c^{0}=T] < c=2$ we're done as by (12) also $c\ < c=2$ (using that $Pr[a \wedge b] \quad Pr[a]$ for all events a; b). To nish the proof of the claim we need to show that otherwise, i.e., if

$$Pr[Y \quad Q\ c^{0}=T] \quad c=2 \qquad (13)$$

then

$$Pr[\quad < k\ ] < c=2 \qquad (14)$$

as this again implies $c\ < c=2$. Eq. (13) (using $Pr[V \quad x] \quad p\ ) \quad E[V] \quad x\ p$ which follows from Markov's inequality) gives

$$E[Y\ ] \quad Q\ c^{0}\ c=2T \qquad (15)$$

Plugging this into (10), then using or choice (2) of $c' = c''=c^2$ and in the last step of $Q \quad q(T) = 5 \quad T^2 \quad c=c'' + 5T$ (this bound for $q(T)$ was just chosen so the last inequality below works out nice)

$$E[\quad] \overset{(10)}{\quad} \frac{c \quad T}{8} \quad \frac{E[Y] \quad T}{Q + T}$$

$$\overset{(15)}{\quad} \frac{c \quad T}{8} \quad \frac{\frac{Q \quad c' \quad c}{2T} \quad T}{Q + T}$$

$$\overset{(2)}{\quad} \frac{Q \quad c'' \quad 2T^2 \quad c}{16(Q + T)}$$

$$c''=32$$

Using the Cherno bound as in Proposition 4 (see Appx. B) with $= 1=2$ and $c'' \quad 256 \ln(c=2)$.

$$\Pr[\quad < c''=64] \quad \Pr[\quad < E[\quad]=2] \quad e^{\quad E[\quad]=8} \quad e^{\quad c''=256} \quad c=2$$

With our choice (2) of $c''= \max f\, 64k;\quad 256 \ln(c=2)g$ we get the bound $\Pr[\quad < k] \quad c=2$ claimed in (14). $\quad 4$

4.3 A technical lemma

Consider any $t; z \ 2 \ N$, $z > t$, $t = 0 \bmod 2$ and $p_1; : : : ; p_z \ 2 \ [0; 1]$. Denote with $p \overset{def}{=} \frac{\sum_{i=1}^{z} p_i}{z}$ be the average value.

Lemma 5. For any $q_1; : : : ; q_z \ 2 \ [0; 1]$, (de ning $q_i = 0$ for $i \quad 0$) and integers $^0; \quad$ where $0 \quad ^0; \quad 0$, if $p_1 = p_2 = : : : = p \quad = 0$ then

$$\sum_{2f \quad ^0; \quad ^0+ \quad g} \sum_{i=1}^{z} jp_i \quad q_i \quad j \quad p \qquad (16)$$

We postpone the proof of the lemma, but let us observe that for example it implies, that if $p_1 = p_2 = : : : = p_t = 0$, then

$$\frac{1}{t} \sum_{=0}^{t} \sum_{i=1}^{z} jp_i \quad q_i \quad j = \frac{1}{t} \sum_{^0=0}^{t=2} \underbrace{\sum_{2f \quad ^0; \quad ^0+ \, t=2g} \sum_{i=1}^{z} jp_i \quad q_i \quad j}_{t \ p=2 \text{ by } (16)} \quad \frac{t}{4} \quad p \qquad (17)$$

Looking ahead, the lhs. of Eq. (17) will denote the expected number of times the master circuit detects an inconsistency in the experiment, whilep denotes the fraction of outputs where F_1 diverts. So if the fraction of wrong outputs is larger than $4=t$, the master circuit will on average raise an alert once. To get a bound on the security the expectednumber of alerts is not relevant, only in the probability that it's larger than one, as this means that a Trojan was detected. The more ne grained statement Eq. (16) will be more useful to argue this.

Fig. 5. Illustration of the main variables used in the proof of Lem. 5.

Proof (of Lem. 5). The lhs of (16) is equal to a sum of $2z$ terms of a form $|p_i - q_j|$ (with $\in \{0, 0+\}$ and $i = 1, \ldots, z$). Let $S_{\ell,0}$ be the multiset consisting of the values of these terms, i.e.,

$$S_{\ell,0} \stackrel{\text{def}}{=} \bigcup_{\in\{0,0+\}} \bigcup_{i=1}^{z} \{|p_i - q_{j-\ell}|\}$$

As an example, the set $S_{2,1}$ is illustrated with the purple boxes in Figure 5, in this figure, the grid point (ℓ, i) corresponds to the value $|p_i - q_{j-\ell}|$. To prove the lemma we need to show that

$$\sum_{s \in S_{\ell,0}} s \le p \tag{18}$$

Figure 5 illustrates the proof idea. Each point (ℓ, i) in the grid corresponds to the value $|p_i - q_{j-\ell}|$, and the values in the two purple boxes are the values in the sum $S_{\ell,0}$ we need to bound. We will cover mutiset $S_{\ell,0}$ by multisets Z_j (for $j = 1, \ldots, z$) defined as the following zig-zag sum

$$Z_j \stackrel{\text{def}}{=} \{|p_j - q_{j-0}|, |p_j - q_{j-0}|, |p_j - q_{j-2}|, |p_j - q_{j-2}|, \ldots, |p_j - q_{j-b}|, |p_j - q_{j-b}|, |p_j - q_{j-(b+1)}|\}$$

where $b = d(j-1) = e_{so1} \le j \le b$ (for an example of a set Z_j see Figure 5). Using the triangle inequality ($\forall a, b, c : |a-b| + |b-c| \ge |a-c|$) repeatedly, we get that

$$\sum_{s \in Z_j} s \le |p_j - p_{j-b}| \le p_j,$$

where the last step comes from the fact that $p_{j-b} = 0$, which holds because $j \le b$. By construction, each $s = |p_i - q_{j-\ell}|$ is contained it at most $z =$

di erent Z_j's (this is because each p_i q_j j is a member of Z_j only for j's such that $i = j \pmod{}$). Using this fact in the rst step below

$$\sum_{s2S_{;}\ 0} s \prod_{i=1}^{z} Z_j \prod_{i=1}^{z} p_j = p$$

as claimed.

5 A 12-redundant scheme $_{12}$

In this section we de ne a scheme $_{12}$ and we will show that the lower bound for achievable security for very simple schemes (shown in Lem. 2) is asymptotically tight. Our proof is constructive - the analysis of our $_{12}$ construction shows that it is $(c; \frac{c^0}{T})$-Trojan resilient for suitable constants.

Our $_{12}$ scheme operates with three independent input streams and one independent bit stream. On each query, every circuit in $_{12}$ receives one of the three inputs and produces an output. The master circuit then checks the consistency of the outputs, i.e. veri es if there is no mismatch between any pair of circuits receiving the same input.

As stated in Sec. 1, digital Trojans mainly employ two types of strategies: time bombs (where time is measured in the number of usages) and cheat codes (as a part of the input). To counter these strategies, $_{12}$ desynchronizes the circuits in two ways. First, some of the circuits are tested in the test phase for a randomly chosen time (already employed in the $_2$ scheme). This e ectively makes it di cult for time bomb Trojans to coordinate the time in which they start deviating. In $_{12}$, half of the circuits are tested for T times where T is a random variable with uniform distribution on $[t]$.

The second method of desynchronization involves using the value of the aforementioned input bit to alternate the way inputs are distributed among the circuits. Consequently, cheat code Trojans are rendered ine ective as only a subset of the circuits share the same input. Moreover, at any given point in time a circuit never knows which alternating state it is in (i.e. it does not know whether its output would be compared with deviating circuits or not).

The main building block of the $_{12}$-scheme is a group of four circuits: two outer ones and two inner ones (see Fig. 6). On each query, every group of circuits receives two inputs - the rst is given to the outer circuit on the left and the second to the outer circuit on the right. Additionally, in every step a fresh decision/alternation bit b is sampled. According to its value these two inputs are given to the inner circuits. $_{12}$ consists of three such groups. Crosschecks are performed whenever two distinct circuits receive the same input (both within a group and among groups).

The proof that the construction $_{12}$ is actually Trojan-resilient starts with assuming that it is not secure, goes via a hybrid argument and leads to a contradiction with security of $_2$ construction. In every hybrid we change the construction slightly by swapping some pairs of circuits, arguing that the advantage of the

Fig. 6. In a given group of circuits, depending on the value of b, the leftmost and rightmost circuits (outer circuits) are paired with the circuits in the middle (inner circuits). Circuits in a pairing are given the same input, and their outputs are checked for equivalence.

adversary does not change much between each successive hybrids. In the final hybrid we show that the modified construction contains $\Pi_2^?$ as a sub-construction. It turns out, that any adversary who wins with reasonable good probability in the final hybrid can be used to build an adversary who breaks the security of $\Pi_2^?$ which is a contradiction with Theorem 2.

5.1 The Π_{12} scheme

We will now define our Π_{12} construction. It is illustrated in Fig. 3 (see page 11). We view our 12-circuit construction as three groups of four circuits each. Group 1 consists of circuits $F_1, \ldots, F_4$, group 2 consists of $F_5, \ldots, F_8$, and group 3 consists of $F_9, \ldots, F_{12}$. At the beginning the three independent and identically distributed sequences of inputs are sampled. Moreover, independent sequence of bits is sampled (it is used to alternate the inputs' distribution in the wild). For every query in the wild, the construction performs two steps: (i) the querying step, where the inputs are distributed to all the 12 circuits depending on the value of the corresponding bit (ii) the cross-checking step where the master circuit checks the consistency of the outputs of the circuits who receive the same inputs.

Now we can take a closer look on our construction. There are three pairs of circuits that share the same input throughout the course of the game regardless of the value of the random bit (see Fig. 3). For instance, the circuit pairs (F_2, F_6), (F_3, F_{11}) and (F_7, F_{10}) share the exact same inputs throughout the game. The outer two circuits within each circuit group (circuits F_i for $i \equiv 0, 1 \bmod 4$) are uniquely paired with exactly one of the middle circuits, i.e. given the same input, depending on the value of the random bit b_i sampled by the master circuit at each step of the game. For instance, in circuit group 1 if $b_i = 0$, F_1 is paired with F_2 and both circuits given x_i^1 as input, and F_4 is paired with F_3 and both given x_i^2 as input. After the querying phase, the master cross-checks the output of the circuits which share the same input streams. If any of the cross checks in any

round fail, then the master aborts and the adversary looses. We now provide a more detailed description of the construction as follows:

test: In the test phase, $T^{F_1, \ldots, F_{12}, F}(T)$ picks a random such that $0 < T < 1$, then queries F_1, F_4, F_5, F_8, F_9 and F_{12} on random and independent inputs $x_i^1, x_i^4, x_i^5, x_i^8, x_i^9$ and x_i^{12} respectively and checks if the outputs of the corresponding circuits are correct by comparing them with the trusted F.

master: The master samples three independent input streams $\vec{x}_1 = (x_1^1, x_2^1, x_3^1, \ldots)$; $\vec{x}_2 = (x_1^2, x_2^2, x_3^2, \ldots)$; $\vec{x}_3 = (x_1^3, x_2^3, x_3^3, \ldots)$ and an independent bit string $\vec{b} = (b_1, b_2, \ldots)$. The operation of the master circuit is split into two phases: (i) querying phase and (ii) cross-checking phase.

Querying step. For all $i \in [Q]$, it queries the functions $F_1, F_2, \ldots, F_{12}$ as follows:

1. If $b_i = 0$,
   The functions F_1, F_2, F_5, F_6 get x_i^1 as input,
   The functions F_3, F_4, F_{11}, F_{12} get x_i^2 as input, and
   The functions F_7, F_8, F_9, F_{10} get x_i^3 as input.
2. if $b_i = 1$,
   The functions F_1, F_3, F_9, F_{11} get x_i^1 as input,
   The functions F_2, F_4, F_6, F_8 get x_i^2 as input, and
   The functions F_5, F_7, F_{10}, F_{12} get x_i^3 as input

Cross-checking step. For all $i \in [Q]$, the the master circuit pairwise compares the outputs of the circuits that receive the same inputs (see Appendix A for the details of the cross-checking phase). If at any round any of the checks fail, the master outputs abort and the adversary looses.

Output. If all the checks succeed in the cross-checking phase, the master outputs the output of the circuit F_1, i.e., $y = F_1(\vec{x}^1)$ as the output of $_{12}$.

5.2 Security of $_{12}$

In this section we prove Thm. 1, which states that the construction presented in Sec. 5.1 is $(c, \frac{c'}{T})$-secure for appropriate choice of constants c and c'. More precisely, we show that the security of the 2-circuit construction from Sec. 2.9 can be reduced to the security of the 12-circuit construction presented above. Before proceeding with the proof, we introduce some useful definitions and notations.

5.2.1 History hardcoded circuits and plaits.

We observe that the notation $F(x)$ for stateful circuits is ambiguous, since its value depends also on the history of queries to F (which is not provided as a parameter). We can thus assume that each F is associated with some stream $\vec{x} = (x_1, x_2, \ldots)$ and that $F(x_i) := F(x_i | x_1, x_2, \ldots, x_{i-1})$. This notation uniquely describes the i-th query to F given the stream $\vec{x}$.

In our proof we will however need a slightly different notion called history-hardcoded circuits. Given any stateful circuit F and two arbitrary streams $\vec{x} = (x_1, x_2, x_3, \ldots)$ and $\vec{w} = (w_1, w_2, w_3, \ldots)$, we say $F^{\vec{x}}$ is an $\vec{x}$-history-hardcoded circuit if at the i-th query it hardcodes the stream values $x_1, \ldots, x_{i-1}$ as its

history and takes w_i from the stream w as the input to query i. Thus F^x on the i-th query with input w_i returns the value: $F^{x,i}(w_i) = F(w_i|x_1, x_2, :::, x_{i-1})$ and on the $i+1$-th query returns $F^{x,i+1}(w_{i+1}) = F(w_{i+1}|x_1, x_2, :::, x_{i-1}, x_i)$. We call the stream x the hardcoded history stream and w the input stream.

For a random variable X which takes values from $\{X_1, X_2, :::\}$ and a circuit F we define another random variable F^X as follows. Its value for $X = x$ is simply F^x. We will call this random variable an X-history-hardcoded circuit. Note that as long as F^X receives inputs from a stream W independent from X, we can say that F^x is a history-independent circuit.

We emphasize that when the hardcoded history stream is equal to the actual input stream, the history-hardcoded circuit returns the same results as the original stateful circuit receiving the same input stream. In other words:

$$F(X_i) = F^{X,i}(X_i); \tag{19}$$

for all $i \in \mathbb{N}$ with probability 1.

Another idea exploited in our construction is the concept of *alternating* inputs depending on the values of random bits. We will express this idea using the notion of b-plaits, where b is a stream of random bits. A b-plait of two streams a^0 and a^1 is a new stream $(a^0a^1)_b$, where its i-th value is either a^0_i from stream a^0 or a^1_i from stream a^1 depending on the i-th value of the decision stream b. More precisely:

$$(a^0a^1)_b = (a^{b_1}_1, a^{b_2}_2, a^{b_3}_3, :::)$$

In our construction, there is only one decision stream used for every plait, therefore the b will be omitted for simplicity. Thus to express the plait of two streams a^0, a^1 we will simply write a^0a^1. A plait of two identical streams of say s will simply be written as s, rather than ss.

Similarly to b-plaits of streams we can define the plaits of history-hardcoded circuits. Let $G^{x^0}_0$ be an x^0-history-hardcoded circuit and $G^{x^1}_1$ be an x^1-history-hardcoded circuit. We say $(G^{x^0}_0 G^{x^1}_1)_b$ is b-plait for $G^{x^0}_0, G^{x^1}_1$ if

$$(G^{x^0}_0 G^{x^1}_1)^i_b(x) = G^{x^{b_i},i}_{b^i}(x): \tag{20}$$

Note that the plaited circuit $(G^{x^0}_0 G^{x^1}_1)_b$ can be expressed as a function of G_0, G_1 and streams x^0, x^1. Looking ahead, this notion of plaited circuits will be crucial in our final reduction of the security of $_{12}$ to $_2^?$.

Finally, we define an operation on history-hardcoded circuits in the context of our construction:

Swap(F^x, G^t) : Given two history hardcoded circuits F^x and G^t in our construction, this operation physically exchanges the positions of both circuits. That is, that F^t physically replaces G^x and vice versa. Swapped circuits keep their histories, but since they change their place in the construction, they now receive potentially different inputs and are crosschecked with different circuits.

An important notion related to the Swap operation which we will exploit in a proof is a *red edge*. We say there is a *red edge* in the k-th query between

two history hardcoded circuits F^x and G^t i after performing the $Swap(F; G)$ operation there is a change in either of the outputs of the swapped circuits on the k-th query compared to the outputs of the circuits if the Swapoperation was not performed. Looking ahead, the notion of swaps and red edges would be used in our proof to show that modifying the original $_{12}$ construction by someSwap operations does not change much the security parameters.

Now, given these de nitions, we are ready to present an intuition that lies behind our construction. We might (and should) ask the authors "but why 12 circuits?". The reason is understandable: it is hard to perform any direct proof for history-dependent circuits; things become too complicated. Fortunately, there exist reductions. As long as we have a valid proof of theorem 2 for history-independent circuits, we can try to nd some construction of history-dependent circuits which can be reduced to it. The main goal is to design the crosschecks is such a way, that, informally speaking, making circuit more history-dependent make the whole construction more secure It is not hard to believe in such a statement; thanks to the alternating random bit, you never know which of some two circuits will receive a speci c input. If these two circuits are very history dependent and have independent histories, there is a high probability, that on the given input they would answer di erently. Thank to crosschecks, the master may detect such inconsistency with high probability. To make a practical advantage of this remark, we need to perform manySwapoperations and analyze the behaviour of various parameters describing our construction. We were able to handle such design and analysis for 12 circuits construction.

Now we will give a more detailed description of the intuition. As written a few lines before, the main idea of the proof is to reduce the construction which consists of (possibly) history-dependent circuits to $^?_2$. $^?_2$ consists of 2 history-independent circuits (alternatively speaking - pairs of circuits with di erent hardcoded histories, independent of the inputs that they receive). The Swapoperation $Swap(F^x; G^t)$ is legit whenever either one of the conditions holds - the circuits F and G are engaged in the cross-checking process as pictured in the Figure 6 (e.g. circuits F_1 and F_4 or circuits F_6 and F_7 in the Figure 7 (Hyb_0) or the circuits received the same inputs before performing any swaps (e.g. circuits F_2 and F_7 swapped in Hyb_2 which are placed at the positions ofF_2 and F_6 from Hyb_0 in the Figure 7). Now, the main idea of the proof is that by performing a series of$Swap$operations on the setting with 3 rows of 4-circuit groups, we are able to end up with a setting Hyb_2 that contains 2 pairs of history-independent circuits at the place of cross-checked circuit pairs (F_1, F_2) and (F_3, F_4). We need just 1 Swapoperation in the middle row to have history-independent circuits in the place of F_1 and F_4, but for F_2 (and F_2) we will need an additional input stream that goes with a new row.
We are now ready to proceed to the proof of Thm. 1.

5.2.2 Proof of Thm. 1. The proof of Thm. 1 proceeds in two parts. We ultimately want to prove a reduction from the security of $_{12}$ to that of $^?_2$. Nevertheless recall in Lem. 4 the security of $^?_2$ crucially depends on history

independent circuits. Thus the first part of our proof constructs a sequence of three hybrids, Hyb_0, Hyb_1, Hyb_2, to get a pair of history independent circuits, F_4^2 and $F_7^3 F_{10}^3$, in the final hybrid. Hyb_0 is the original construction. To get from Hyb_0 to Hyb_1, we perform the Swap operation on the following pairs of circuits in Hyb_0: $(F_1^1; F_4^2)$; $(F_6^{12}; F_7^3)$; $(F_{10}^3; F_{11}^{21})$. To get from Hyb_1 to Hyb_2, we perform the Swap operation on the following pairs of circuits in Hyb_1: $(F_2^{12}; F_7^3)$; $(F_3^{21}; F_{10}^3)$ (refer to Fig. 7). Note that in the final hybrid Hyb_2, it is crucial that F_4^2 and $F_7^3 F_{10}^3$ are not just history independent, but also take in the same inputs from input stream 1 regardless of the value of the random bit ($F_7^3 F_{10}^3$ takes inputs from stream 1 due to the definition of plaited circuit in (20)). This will be necessary for the second part of our proof which uses F_4^2 and $F_7^3 F_{10}^3$ in the final hybrid as the two history independent circuits needed for the $\mathcal{E}_2^?$ construction and uses the $\mathcal{E}_2^?$ construction with these circuits as a subroutine.

Fig. 7. Hybrids with the circuits and their corresponding plaited hardcoded history and input streams (above and below each circuit in black respectively). In Hyb_2, F_4^2 and the plaited circuit $F_7^3 F_{10}^3$ (in red) are history independent.

Proof. For a given $F_1, \ldots, F_{12}$, we can define some random variables as follows. Let $\mathrm{F}_{j}^{A,B}$ be the total number of queries, where F_j^A gets input from a stream B and has a mismatch with any other circuit getting input from the same stream in this query. We will refer to random variables related to the i-th hybrid by adding a superscript i. For example, $\mathrm{F}_{1}^{0,1;2} = 0$, since in Hyb_0 no crosschecks are made between F_1^1 and the circuits receiving inputs from stream 2. Let bad be the total number of mismatches detected by the master circuit. Recall from Sec. 2.3 that Y is the total number of mistakes the master circuit makes. The probability space of these random variables is the set of all choices of a stream of random bits b and streams of random inputs $1, 2, 3$ and a number of tests T.

We prove our statement by contradiction. To this end, we assume that

$$\exists_{c>0} \forall_{c'>0; T_0 \in \mathbb{N}; q \in \mathrm{poly}} \exists_{T > T_0; Q > q(T)} \exists_{\mathrm{Adv}} \text{ such that}$$
$$\mathrm{Adv}(c; \tfrac{c'}{T}) \quad \text{wins} \quad \mathrm{TrojanGame}(\mathcal{E}_{12}; T; Q) \tag{21}$$

Therefore for some c and for all c' there exists an infinite set $T \subseteq \mathbb{N}$ such that for every $t \in T$ there exists an infinite set $Q_t \subseteq \mathbb{N}$ such that for every $t \in T, z \in Q_t$ there exists an adversary $Adv = Adv(c, c', z, t)$ such that the following formula is true:

$$\Pr[\ldots] > \frac{z}{t^c} \quad (22)$$

Now we will look what happens to inequality (22) as we move through each hybrid:

Hyb_0 : Hybrid 0 corresponds to the original construction due to equality (19). Hence, the probability that the adversary $Adv(c, c', z, t)$ wins in this hybrid is precisely that in Equation (22).

Hyb_1: In this hybrid we simply perform three Swap operations on the following pairs of circuits: (F_1^1, F_4^2); (F_6^{12}, F_7^3); (F_{10}^3, F_{11}^{21}).

Claim. $\Pr[\ldots] \leq \frac{1}{F_4^2;1}; \frac{1}{F_7^3;12}; \frac{1}{F_{10}^3;21}$ $k \wedge \ldots] \leq \frac{z}{t^c}$ $3k \cdot 2^{-k}$.

Proof. After the first two swaps two disjoint events may occur, either the number of the red edges exceeds some value k or it stays small. We study these two cases below:

> $Huge_k$ - any number of red edges between (F_1^1, F_4^2), (F_6^{12}, F_7^3), (F_{10}^3, F_{11}^{21}) is greater than k. We would like to show, that
>
> $$\Pr[Huge_k \wedge (\ldots)] \leq 3 \cdot 2^{-k}. \quad (23)$$
>
> We will first consider the red edges between F_1^1, F_4^2. For this purpose we define an event $huge_k$ indicating that the number of red edges between F_1^1, F_4^2 is greater than k. We consider the following game that shows that in this case the the Adv is easily caught on cheating. G lasts for z steps with a player P: at the beginning streams $1^0, 2^0, b$ are sampled uniformly and independently from $X^z, X^z, \{0,1\}^z$, respectively. The i-th step has the following substeps:
>
> 1. $1_i^0, 2_i^0$ are sampled uniformly and independently from X and revealed to P.
> 2. P decides if this step is *red*, i.e. if she wants to make a guess.
> 3. If this step is red, P makes a guess, i.e. outputs a bit d_i.
> 4. b_i is sampled uniformly and independently from $\{0,1\}$ and revealed to P.
> 5. If P made a guess, we say P is correct if $d_i = b_i$.
>
> P wins, if she made at least k guesses and all of them were correct. Obviously the probability of winning is not greater than 2^{-k} for any P. On the other hand, we can analyze, what happens, if the adversary employs the circuits F_1, F_4 from the original experiment to play the game for him.

The strategy of the player P in game G is following: $F_2^{12}; F_3^{21}$ in the i-th step get inputs $1_i^0; 2_i^0$, respectively. The step is red, if there is a red edge in the original experiment, i.e. if:

$$F_1^1(1_i^0) \neq F_4^2(1_i^0) \vee F_1^1(2_i^0) \neq F_4^2(2_i^0):$$

We do not analyze the way d_i is chosen, since $\Pr[d_i = b_i] = \frac{1}{2}$: For such a strategy P wins the game with probability at least $\Pr[huge_k \wedge (\ ^0 = 0)]$, therefore

$$\Pr[huge_k \wedge (\ ^0 = 0)] \quad 2^{\ k}:$$

The same conclusion works for pairs $(F_6^{12}; F_7^3); (F_{10}^3; F_{11}^{21})$, therefore by the bound on the sum of the probabilities we can conclude that the inequality (23) holds.

$Tiny_k$ - in this case every number of red edges between $(F_1^1; F_4^2), (F_6^{12}; F_7^3); (F_{10}^3; F_{11}^{21})$ is not greater than k. 3 Swap operations performed in Hyb_1 will not change the value Y^1 by more than $3k$ and the behavior of e.g. F_4^2 differs from F_1^1 only on up to k queries, i.e.

$$\frac{1}{F_4^2;1}; \frac{1}{F_7^3;12}; \frac{1}{F_{10}^3;21} \quad k \tag{24}$$

$$jY^0 \quad Y^1 j \quad 3k: \tag{25}$$

Now we can transform inequality (22)

$$\begin{aligned} c \quad 3 \quad 2^{\ k} \quad \Pr[\ ^0 = 0 \wedge Y^0 \quad c^0(\tfrac{z}{t})] \quad 2^{\ (k+1)} = \\ = \Pr[Tiny_k \wedge \ ^0 = 0 \wedge Y^0 \quad c^0(\tfrac{z}{t})] + \\ + \Pr[Huge_k \wedge \ ^0 = 0 \wedge Y^0 \quad c^0(\tfrac{z}{t})] \quad 2^{\ (k+1)} \\ \Pr[Tiny_k \wedge \ ^0 = 0 \wedge Y^0 \quad c^0(\tfrac{z}{t})] \\ \Pr[\frac{1}{F_4^2;1}; \frac{1}{F_7^3;12}; \frac{1}{F_{10}^3;21} \quad k \wedge \ ^0 = 0 \wedge Y^1 \quad c^0(\tfrac{z}{t}) \quad 3k] : \end{aligned}$$

4

Hyb_2 : In this hybrid we simply perform two Swap operations on the following pairs of circuits: $(F_2^{12}; F_7^3); (F_3^{21}; F_{10}^3)$.

Claim.

$$\Pr[(\ _{F_4^2;1}^2 \quad 3k) \wedge (Y^2 \quad c^0(\frac{z}{t}) \quad 5k)] \quad c \quad 3 \quad 2^{\ k}: \tag{26}$$

Proof. Every Swap operation performed in Hyb_2 changes the value of $Y^2; \ _{F_4^2;1}^2$ by at most k (since inequality (24) holds). The inequality is explicit. 4

Claim. For every $k \in N$ and every adversary Adv who $(c; \frac{c^0}{t})$-wins $(\ _{12}; T; Q)$ TrojanGame there exists an adversary Adv^0 who $(c \quad 3 \quad 2^{\ k}; \frac{c^0}{t} \quad \frac{5k}{z})$-$(3k+1)$-wins the game TrojanGame $(\ _2^?; T; Q)$.

We want to conclude, that the above statement contradicts Lem. 4. So we want to show, that this incorrect statement is implied by our construction.

$$\exists_{\epsilon>0}\forall_{c'>0,T_0\in\mathbb{N},q\in poly}\exists_{T>T_0,Q>q(T)}\exists_{Adv}\text{ such that}$$
$$Adv(\epsilon,\tfrac{c'}{T})\text{ wins }TrojanGame(\mathcal{F}_2^?,T,Q). \tag{27}$$

Let $k = 2+\log(\frac{1}{c})$ and $\epsilon = c\cdot 3\cdot 2^{-k} = \frac{c}{4} > 0$. Choose c' arbitrarily and let $c' = \tilde{c}'$. Let $\tilde{T} = T$. Let

$$\tilde{Q}_t = \{z\in Q_t : z > t\cdot(\frac{5k}{\tilde{c}'}+1)\}.$$

Obviously $\tilde{Q}_t$ is infinite. As a result, for every $q\in poly$ there exists $z\in\tilde{Q}_t$ such that $z > q(t)$.

Now we can construct the adversary $\tilde{A}dv$ which would break the security of $\mathcal{F}_2^?$ which lead us to contradiction. Thanks to the analysis of the hybrids we know, that for Adv the inequality (26) holds. Define the circuits $\tilde{F}_1,\tilde{F}_2$ given to $\tilde{A}dv$ in the following way:

$$\tilde{F}_1 = F_4^2,\quad \tilde{F}_2 = F_7^3F_{10}^3,$$

where the latter is a b-plait (as defined in Equation (20)). Actual values of streams $2,3,b$ are sampled uniformly and independently by $\tilde{A}dv$, and hardcoded in $\tilde{F}_1,\tilde{F}_2$. Obviously $\tilde{F}_1,\tilde{F}_2$ are history independent, therefore $\tilde{A}dv$ meets the requirements for the $\mathcal{F}_2^?$ scheme.

Now we can bound a random variable e - the number of queries in a $(\mathcal{F}_2^?,T,Q)$ $TrojanGame$ where the adversary is caught on deviating. If $\mathcal{F}_{F_4^2,1}^2\le 3k$, then $e\le 3k$, since if in the i-th query there was any inconsistency between $\tilde{F}_1,\tilde{F}_2$, there must had been a mismatch between F_4^3 and any other circuit receiving the same input. Which concludes in:

$$\begin{aligned}\epsilon = c\cdot 3\cdot 2^{-k} &\le \Pr\left[(\mathcal{F}_{F_4^2,1}^2\le 3k)\wedge(Y^2\le c'(\tfrac{z}{t}))\right]\le 5k)\\ &\le \Pr\left[e\le 3k\wedge(Y^2\le c'(\tfrac{z}{t}))\le 5k\right]\\ &\le \Pr\left[e\le 3k\wedge(Y^2\le \tilde{c}'(\tfrac{5k}{\tilde{c}'}+1))\le 5k=\tilde{c}'\right]:\end{aligned}$$

Note that Y^2 is the number of mistakes made by the master circuit in the $TrojanGame(\mathcal{F}_2^?,T,Q)$ and the last inequality comes from $z > t\cdot(\frac{5k}{\tilde{c}'}+1)$. We conclude, that that there exists ϵ, such that for every c' there exists $\tilde{A}dv$ who $(\epsilon,\frac{c'}{t})$-$(3k+1)$-wins $TrojanGame(\mathcal{F}_2^?,T,Q)$. It is with contradiction with Lem. 4, which ends the proof.

5.3 Reapplying the hybrid argument

In the previous section, we have used the outputs produced on the input stream 1 in place of F_1 as an output stream of the construction. By symmetry, the

argument from the previous section works for the input stream 2 in place of F_4. Now we will show that in fact the outputs from F_5 or F_8 may be used as an output stream of the construction, which also implies the possibility of using input stream 3 to produce the output stream of the construction.

Now, in Hyb_0 (Figure 7) firstly swapping the labels of the input streams 3 and 2 on the input bit 0 and the labels of the input streams 3 and 1 on the input bit 1 and secondly visually swapping the rows 1 and 2 does not change the setting. We achieve the Hyb_0^{00} construction as shown in Fig. 8.

Fig. 8. Hyb_0^{00} construction.

We can still reapply the hybrid argument from the previous section to the modified Hyb_0^{00} construction by applying the following swaps. $\mathrm{Hyb}_0^{00} \to \mathrm{Hyb}_1^{00}$ swaps: $(F_5^1; F_8^2)$; $(F_2^{12}; F_3^3)$; $(F_{11}^3; F_{10}^{21})$. $\mathrm{Hyb}_1^{00} \to \mathrm{Hyb}_2^{00}$ swaps: $(F_3^{12}; F_6^3)$; $(F_7^{21}; F_{11}^3)$. Finally we conclude that the output streams of either F_5 or F_8 (by symmetry) may be used as an output stream of the construction, what implies that by taking outputs from circuit F_8 on input bit 0 and circuit F_5 on input bit 1 could be used as an output of the construction produced with input stream 3.

The above argument implies that in each round our construction may output 3 outputs produced by consuming inputs from the same number of 3 input streams.

6 Outlook and Open Problems

In this work we introduced countermeasures against hardware Trojans which compared to existing solutions based on multiparty or verifiable computation are extremely simple and efficient, but only achieve limited security guarantees (i.e., we only guarantee that most outputs are correct or the Trojan is detected with high probability) in a restricted setting (iid inputs and no evolving state).

Because of this, the scope of application for our schemes is limited, but as discussed in the introduction, we believe they will serve as a first but major step towards solving some of the main application targets like randomness generation. In particular, creating pseudorandomness for randomness hungry side-channel

countermeasures like masking in a Trojan-resilient way is one of the main motivations for this work. The reason our simple schemes are a promising starting point towards Trojan-resilient pseudorandomness generation is the fact that in most settings (like masking) one does not need that the pseudorandomness is correctly generated, only that it is indistinguishable from uniform, so the relaxed security of our schemes is not a deal breaker. Another reason is the fact that one could use some of the pseudorandomness that is created to implement the master's randomness source, thus making it deterministic. Fleshing these ideas out will require a better understanding of ampli cation and circularity issues in this setting.

The main concrete technical question left open problem in this work is to prove the security of the minimal and thus really practical scheme $_2$ as stated in Conjecture 1. A positive resolution of the conjecture will need techniques that go beyond our proof via history-independence used in the proof for $_{12}$.

References

1. Adee, S.: The hunt for the kill switch, spectrum, https://tinyurl.com/j95zbmxa
2. Ateniese, G., Kiayias, A., Magri, B., Tselekounis, Y., Venturi, D.: Secure outsourcing of cryptographic circuits manufacturing. In: Baek, J., Susilo, W., Kim, J. (eds.) ProvSec. Lecture Notes in Computer Science, vol. 11192, pp. 75 93. Springer (2018). https://doi.org/10.1007/978-3-030-01446-9_5, https://doi.org/10.1007/978-3-030-01446-9_5
3. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) ITCS. pp. 326 349. ACM (2012). https://doi.org/10.1145/2090236.2090263
4. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015), http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053
5. Dubhashi, D.P., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press (2009), http://www.cambridge.org/gb/knowledge/isbn/item2327542/
6. Dziembowski, S., Faust, S., Standaert, F.: Private circuits III: hardware trojan-resilience via testing ampli cation. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS. pp. 142 153. ACM (2016). https://doi.org/10.1145/2976749.2978419
7. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS. pp. 293 302. IEEE Computer Society (2008). https://doi.org/10.1109/FOCS.2008.56
8. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC. Lecture Notes in Computer Science, vol. 2951, pp. 258 277. Springer (2004). https://doi.org/10.1007/978-3-540-24638-1_15, https://doi.org/10.1007/978-3-540-24638-1_15

9. Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_27, https://doi.org/10.1007/978-3-540-45146-4_27
10. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5_9, https://doi.org/10.1007/3-540-68697-5_9
11. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25
12. Maurer, U.M., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4622, pp. 130–149. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_8, https://doi.org/10.1007/978-3-540-74143-5_8
13. Mitra, S., Wong, H.S.P., Wong, S.: Stopping hardware trojans in their tracks, spectrum, https://tinyurl.com/5emst8f2
14. Mukhopadhyay, D., Chakraborty, R.S.: Hardware Security: Design, Threats, and Safeguards. Chapman & Hall/CRC, 1st edn. (2014)
15. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 462–482. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_27, https://doi.org/10.1007/978-3-642-01001-9_27
16. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_26, https://doi.org/10.1007/978-3-642-01001-9_26
17. Tehranipoor, M., Salmani, H., Zhang, X.: Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection. Springer Publishing Company, Incorporated (2013)
18. Wahby, R.S., Howald, M., Garg, S., Shelat, A., Walfish, M.: Verifiable asics. In: IEEE SP. pp. 759–778. IEEE Computer Society (2016). https://doi.org/10.1109/SP.2016.51

A Missing Details of the $_{12}$ Construction

Cross-checking phase We present the details of the cross-checking phase of our $_{12}$ construction as follows:

- if $b_i = 0$, check if
  $\mathsf{F}_1(x_i^1) \stackrel{?}{=} \mathsf{F}_2(x_i^1)$, $\mathsf{F}_5(x_i^1) \stackrel{?}{=} \mathsf{F}_6(x_i^1)$, and $\mathsf{F}_2(x_i^1) \stackrel{?}{=} \mathsf{F}_6(x_i^1)$.
  $\mathsf{F}_3(x_i^2) \stackrel{?}{=} \mathsf{F}_4(x_i^2)$, $\mathsf{F}_{11}(x_i^2) \stackrel{?}{=} \mathsf{F}_{12}(x_i^2)$, and $\mathsf{F}_3(x_i^2) \stackrel{?}{=} \mathsf{F}_{11}(x_i^2)$,

$F_7(x_i^3) \stackrel{?}{=} F_8(x_i^3)$, $F_9(x_i^3) \stackrel{?}{=} F_{10}(x_i^3)$, and $F_7(x_i^3) \stackrel{?}{=} F_{10}(x_i^3)$,

– if $b_i = 1$, check if

$F_1(x_1^i) \stackrel{?}{=} F_3(x_i^1)$, $F_9(x_i^1) \stackrel{?}{=} F_{11}(x_i^1)$, and $F_3(x_i^1) \stackrel{?}{=} F_{11}(x_i^1)$.

$F_2(x_i^2) \stackrel{?}{=} F_4(x_i^2)$, $F_6(x_i^2) \stackrel{?}{=} F_8(x_i^2)$, and $F_2(x_i^2) \stackrel{?}{=} F_6(x_i^2)$,

$F_5(x_i^3) \stackrel{?}{=} F_7(x_i^3)$, $F_{10}(x_i^3) \stackrel{?}{=} F_{12}(x_i^3)$, and $F_7(x_i^3) \stackrel{?}{=} F_{10}(x_i^3)$,

B Chernoff Bound

Proposition 4 (Cherno (see, e.g., [5])). *Let* $X = \sum_{i=1}^{n} X_i$ *where the* X_i *are independent random variables in* $\{0;1\}$. *Then for any* $0 < < 1$ *we have* $\Pr[X < E[X](1 \quad)] \quad e^{-E[X]\epsilon^2/2}$.