Cryptographic Tamper Evidence

February 12, 2003

Gene Itkis

Boston University Computer Science Dept.
111 Cummington St.
Boston, MA 02215, USA
itkis@bu.edu

Abstract. We propose a new notion of *cryptographic tamper evidence*. A tamper-evident signature scheme provides an additional procedure Div which detects tampering: given two signatures, Div can determine whether one of them was generated by the forger. Surprisingly, this is possible even after the adversary has inconspicuously learned (exposed[1]) some — or even *all* — the secrets in the system. In this case, it might be impossible to tell which signature is generated by the legitimate signer and which by the forger. But at least the fact of the tampering will be made evident.

We define several variants of tamper-evidence, differing in their power to detect tampering. In all of these, we assume an equally powerful adversary: she *adaptively* controls all the inputs to the legitimate signer (i.e., all messages to be signed and their timing), and observes all his outputs; she can also adaptively expose *all the secrets* at arbitrary times.

We provide tamper-evident schemes for all the variants and prove their optimality.

Achieving the strongest tamper evidence turns out to be provably expensive. However, we define a somewhat weaker, but still practical, variant: α-*synchronous* tamper-evidence (α-te) and provide α-te schemes with logarithmic cost. Our α-te schemes use a combinatorial construction of α-separating sets, which might be of independent interest.

We stress that our mechanisms are purely cryptographic: the tamper-detection algorithm Div is stateless and takes no inputs except the two signatures (in particular, it keeps no logs), we use no infrastructure (or other ways to conceal additional secrets), and we use no hardware properties (except those implied by the standard cryptographic assumptions, such as random number generators).

Our constructions are based on arbitrary ordinary signature schemes and do not require random oracles.

1 Introduction

Key exposure is a well-known threat for any cryptographic tool. For signatures, exposed keys are revoked *after the exposure is detected.* This detection of the exposure has previously been dealt with outside the scope of cryptography (e.g., delegated to hardware and/or heuristic "forensics"). Indeed, if an adversary inconspicuously learned all the secret information within the system, it may seem that the cryptographic tools without any remaining secrets are helpless against her.

This paper challenges this perception, by providing a cryptographic mechanism to detect the adversary's presence within the system even after she has learned all the secrets. Thus, while it still might not be possible to distinguish forger-generated signatures from the legitimate ones, our new mechanisms can at least make the tampering evident.

1.1 Related work

KEY EXPOSURES: AVOIDANCE AND DAMAGE CONTAINMENT. Some mechanisms to minimize the damage from break-ins have been proposed in the past. A representative sample of these methods

[1] We say that a secret is *exposed* when it becomes known to the adversary. Exposure does *not* imply that the secrets become publicly known. Moreover, nobody — except the adversary — is aware of the exposure taking place.

and references include threshold [DF89,Ped91,BF97], pro-active [OY91,HJJ+97,CHH00], remotely-keyed [Bla95,Bla96,Luc97,BFN98], key-insulated [DKXY02,DKXY], intrusion-resilient [IR02,Itk02], all-or-nothing protection [CDH+00], etc. In these methods, secrets are typically protected by being distributed (shared among multiple modules), thus minimizing the effect of partial exposures.

In this paper, however, we focus on the total and inconspicuous exposures of *all* the secrets within the system at the time of the exposure. For such exposures, only forward-security has being defined [And97,BM99] and achieved [And97,BM99,Kra00,AR00,IR01,MMM02,Itk02,KR02] previously. None of these approaches provided any help in detecting whether an exposure occurred.

Fail-stop signatures. Tamper-evident signatures should be distinguished from the fail-stop signatures [PP97]: the fail-stop signatures do not address the issue of dealing with an adversary who learned the signer's secrets.[2] Instead, they help only in the case of a computationally powerful adversary. Namely, in the fail-stop model, each public key has a large number of valid private key values corresponding to it. An adversary may be powerful enough to compute all of these private keys, but still cannot determine which of these keys is known to the signer. Given a signature forged by such an adversary, the signer, however, can repudiate it by proving that he does not know the specific private key that must have been used to forge the signature. Thus, that approach too does not offer any help in the case when the signer's keys have been exposed.

Coercive exposures. Some previous work addresses the issue of dealing with the situations where the secret keys are exposed under some coercive methods. In such cases, the coerced signer may use some kind of subliminal communication embedded in the signature to inform the authorities that the signature and/or secret keys were extracted from the signer under duress (see, e.g., [HJJY00]).

Another approach proposes *monotone signature schemes* [NPT01], which allow the verification algorithm to be updated after an attack. Namely, under duress the signer can reveal some (but not all!) of his secrets to the adversary. These secrets enable the adversary to generate signatures that are valid according to the *current* verification algorithm. However, this verification algorithm can then be updated so that all the signatures generated by the legitimate signer (before or after the update) remain valid, but all the signatures generated by the adversary are not valid under the updated verification. Again, this approach does not work in case of total exposures. Also, the cost of these signatures is linear in the number of updates. This matches our least efficient (but strongest) construction, except that we do not require the updates of the verification algorithm. In fact, our general lower-bounds proof in Sec. 4 can be modified to yield the linear lower bounds for the length of the monotone signatures[3], thus proving optimality of the results of [NPT01]. To the best of our knowledge no lower bounds for monotone signatures were shown previously. However, slightly modifying the definition of the monotone signatures — to allow key evolution — allows exponential performance improvement (see Sec. 5).

1.2 Our contribution: Tamper Evidence

In contrast to the previous work, we consider the situation where the adversary *inconspicuously* learns *all* secrets of the system at some (unknown) points of time. Our goal is to provide security after such undetected total exposures.

[2] In fact, in [PP97], the authors write "Naturally, this possibility of distinguishing forged signatures from authentic signatures only exists as long as forger has not stolen the signer's key". We show that this observation does not fully apply to the key-evolving signatures (which were introduced after the above paper).

[3] Indeed, some of our constructions (namely, the strongly tamper-evident schemes) bear similarity to some of those of [NPT01]. It may be interesting to explore whether/how some of the optimizations proposed there for the monotone signatures might be applied to improve performance of our schemes (up to constant factors).

INTUITION. Suppose the adversary learns all the secrets at time t_e. Then, clearly, she has all the keys of the legitimate signer, and thus can generate valid signatures. However, if both the signer and the adversary remain active (i.e., generate signatures) then the system has changed fundamentally: instead of having a single entity — the legitimate signer — it now contains two active "versions" running at the same time. Now, if the signer evolves in some randomized fashion, then the two versions will diverge (see Fig. 1), and this divergence might be detectable.

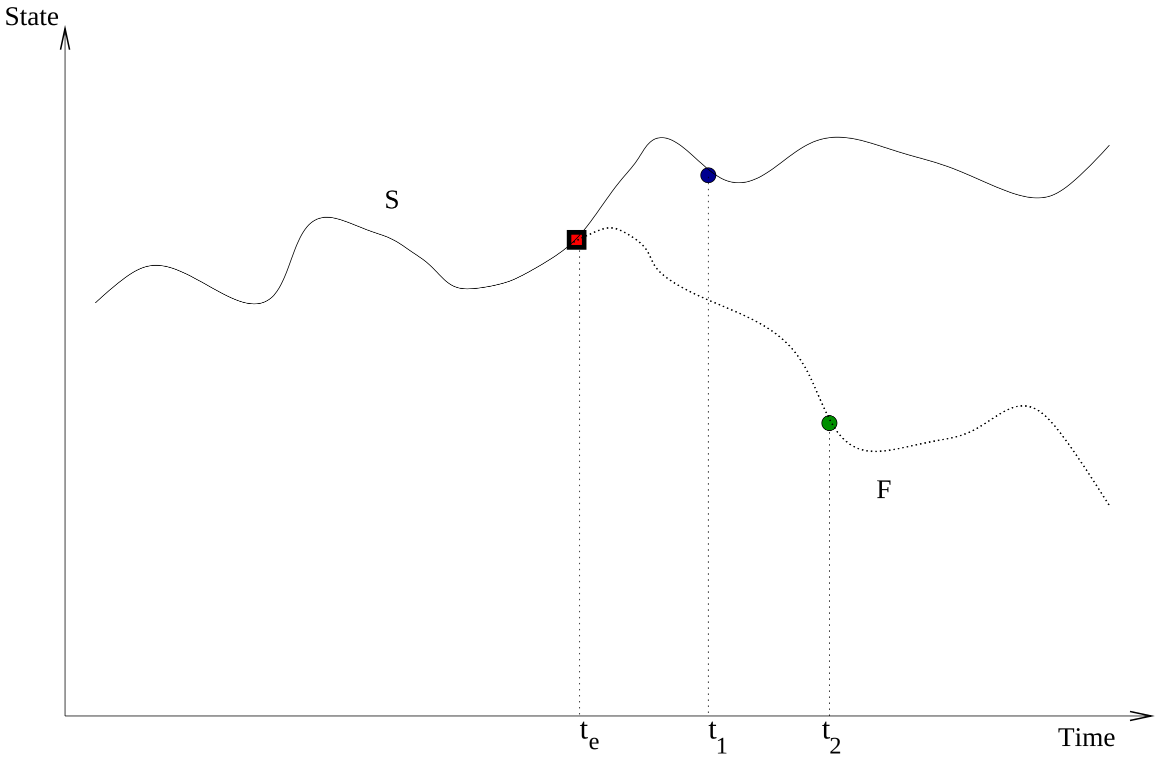

Fig. 1. *Divergence of forger and signer. Signer's secrets are exposed at time t_e. It still may be possible to tell whether signatures, generated at $t_1, t_2 > t_e$, originated from the different "branches" (one signer's, the other – forger's).*

APPROACH. Indeed, this is exactly what we capture in our definitions. We use key-evolving schemes (defined originally for forward-security [BM99]) as the basis for our definitions. Any direct connection between tamper-evident and forward-secure signatures stops at that. In particular, it is crucial for the tamper-evident schemes to use true randomness for the evolution. Even pseudo-randomness is not sufficient, as the seed might be exposed too. But for forward-secure signatures, randomness — even if used, as in [BM99] — can be replaced with pseudo-randomness (as done in [Kra00] to achieve some optimizations). Using true randomness in key evolution allows us to achieve and then detect divergence of the signer's and forger's versions. Detection of this divergence is exactly what constitutes tamper-evidence — thus the name Div for the new procedure.

VARIANTS, CONSTRUCTIONS, LOWER BOUNDS. We define several variants of tamper-evidence. In all the variants we allow the forger F to *adaptively* determine all the messages to be signed by the legitimate signer S, and the times when they are to be signed, before and/or after the exposure and/or the forgery.

The strongest variant guarantees to detect divergence given any two signatures generated by S and F at *any* two time periods after the exposure (it is impossible to do anything beyond forward-security for the periods before the exposure[4]). We present a tamper-evident signature scheme with this strong tamper-evidence property. This scheme imposes linear in time performance penalty. Moreover, we prove that no better scheme is possible.

Fortunately, more efficient schemes are possible for slightly weaker notions of tamper-evidence:

Perfectly-synchronous tamper-evidence works with the same powerful adversary, but guarantees to detect divergence only when the two given signatures are generated by the signer S and the forger F at the *same* time period after the key exposure.

α-synchronous tamper-evidence generalizes the above notions: for such schemes, the tampering is guaranteed to be detected as long as the time periods of the two signatures are relatively closer to each other than to the exposure time. The exact relative proximity is characterized by the parameter α ($\alpha = 0$ yielding strong and $\alpha = \infty$ — perfectly-synchronous tamper-evidence).

We present both perfectly- and α- synchronous tamper-evident schemes. The cost of the first one is only twice that of an ordinary signature scheme. For any finite constant $\alpha > 0$, we construct an α-synchronous tamper-evident scheme with a logarithmic factor overhead (with α appearing in the base of the logarithm). We prove asymptotic optimality of all these schemes.

Next, in Section 2 we formally define tamper-evident signature schemes and present our constructions in Section 3. In Section 4 we prove the optimality of our constructions by proving the information-theoretic lower-bounds. Finally, in the Section 5 we discuss various aspects of the tamper-evident signatures, including their potential applications. We also propose there some improvements for the monotone signatures.

2 Definitions

2.1 Functional definitions

KEY EVOLVING SIGNATURE SCHEMES. As discussed above, tamper-evident signatures must evolve the signer's state, similarly to the forward-secure signatures, but for a different reason.[5] We therefore use the definition of the key-evolving signature schemes proposed by Bellare and Miner [BM99]. Intuitively, in key-evolving schemes the public key remains unchanged, while the corresponding secret key changes periodically. This definition is purely functional: security is addressed separately.

Key-evolving signature scheme is a quadruple of algorithms $\mathsf{KESig} = (\mathsf{Gen}, \mathsf{Upd}, \mathsf{Sign}, \mathsf{Vf})$, where:

$\mathsf{KESig.Gen}$, the probabilistic *key generation* algorithm.
  Input: a security parameter $k \in \mathsf{N}$ (given in unary as 1^k); and the total number of periods T;
  Output: a pair (SK_0, PK), the initial secret key and the public key;

$\mathsf{KESig.Upd}$, the probabilistic *secret key update* algorithm.
  Input: the secret key SK_t for the current period $t < T$;
  Output: the new secret key SK_{t+1} for the next period $t + 1$.

$\mathsf{KESig.Sign}$, the (possibly probabilistic) *signing* algorithm.
  Input: the secret key $SK_t = \langle S_t, t, T\rangle$ for the time period $t \le T$ and the message M to be signed;
  Output: the signature $\langle t, sig\rangle$ of M for time period t.

[4] We can limit the window of vulnerability to the period of the exposure by combining tamper-evidence with forward-security. This window of vulnerability is even stronger if tamper-evidence is combined with intrusion-resilience.

[5] In particular, randomness of the evolution was optional for forward-security, but is crucial for tamper-evidence; while one-wayness of the evolution was of central value for forward-security, but is optional for tamper-evidence.

$\mathsf{KESig.Vf}$, the *verification* algorithm.
Input: the public key PK; a message M; and an alleged signature $\langle t, sig\rangle$;
Output: **valid** if $\langle t, sig\rangle$ is a *valid* signature of M, or **fail** otherwise.

We require that $\mathsf{KESig.Vf}_{PK}(M, \mathsf{KESig.Sign}_{SK_t}(M)) = \textbf{valid}$ for all M and t. For some schemes, T is optional: i.e., $T = \infty$ [Itk02].

DIVERGENCE TEST. As discussed in the introduction, at the core of our tamper-evidence is the observation that after the key exposure there in essence exist two versions of the signer within the system — while under normal conditions (without compromises) there should be only one. Thus we say that the exposure leads to *divergence*. To accommodate functionally the test of this condition — which provides tamper-evidence — we add one more procedure to the above:

$\mathsf{KESig.Div}$, the (possibly probabilistic) *divergence test* algorithm.
Input: two signatures $\langle t_1, \sigma_1\rangle, \langle t_2, \sigma_2\rangle$;
Output: **foul** if divergence is detected; **ok** otherwise.

2.2 Security definitions

SIGNATURE SECURITY. For the sake of brevity, we skip the signature security definition — it is essentially the same as the classic definition of Goldwasser, Micali and Rivest's [GMR88] for (ordinary) digital signatures secure against adaptive chosen message attacks (but as in the other key-evolving schemes — such as forward-secure signatures — the authenticity includes the period number: we consider the adversary successful even if she generates a signature differing only in the period number from one of those generated by the legitimate signer).

TAMPER-EVIDENCE. Say that KESig is *self-consistent* if $\mathsf{KESig.Div}(\langle t_1, \sigma_1\rangle, \langle t_2, \sigma_2\rangle) = \textbf{ok}$ for all signature pairs $\langle t_1, \sigma_1\rangle, \langle t_2, \sigma_2\rangle$, provided that both signatures are generated by the same legitimate signer S (legitimate signer does not deviate from the algorithms specified by the scheme). We consider only self-consistent schemes in this paper.

Definition 1 (Adversary). *Let F be an adversary and S the legitimate signer (for the given instance of* KESig*, generated independently of F). Allow F to adaptively obtain from S both signatures for any time-period/message pairs (t, M),* and *secret keys SK_i. Let t_e be the latest exposure time period: maximum t such that F obtained SK_t. Eventually (after polynomially-bounded time), F must output two signatures $\{\langle t_1, \sigma_1\rangle, \langle t_2, \sigma_2\rangle\}$, such that $t_1, t_2 > t_e$, and $\langle t_1, \sigma_1\rangle$ was generated by S (upon F's request), while $\langle t_2, \sigma_2\rangle$ by F (i.e., the corresponding $\langle t_2, M\rangle$ was not queried from S). We write $F^S \rightarrow \langle t_e, \{(t_1, \sigma_1), (t_2, \sigma_2)\}\rangle$. The probability that the adversary succeeds is* $\mathbf{PrSucc}_{\mathsf{KESig}}(F) \stackrel{def}{=} Prob[\mathsf{KESig.Div}(\langle t_1, \sigma_1\rangle, \langle t_2, \sigma_2\rangle) = \textbf{ok}\ \textit{and}\ \mathsf{KESig.Vf}(\langle t_i, \sigma_i\rangle) = \textbf{valid}, i = 1, 2]$.

Definition 2 (Tamper-Evidence Safety).
Let KESig *be self-consistent, and let k be a security parameter. Let $F^S \rightarrow \langle t_e, \{(t_1, \sigma_1), (t_2, \sigma_2)\}\rangle$. $\{t'_1, t'_2\}$ are t'_e*-safe (for KESig) *if* $\mathbf{PrSucc}_{\mathsf{KESig}}(F) < 1/2^k$ *whenever $t_e = t'_e$ and $\{t_1, t_2\} = \{t'_1, t'_2\}$.*

In other words, let $\mathcal{S}$ be the set of all triplets $\langle t'_e, \{t'_1, t'_2\}\rangle$ such that if for any F as above $F^S \rightarrow \langle t_e, \{(t_1, \sigma_1), (t_2, \sigma_2)\}\rangle$ and $\langle t_e, \{t_1, t_2\}\rangle \in \mathcal{S}$ then $\mathbf{PrSucc}_{\mathsf{KESig}}(F) < 1/2^k$. $\{t'_1, t'_2\}$ are *t'_e-safe* iff $\langle t'_e, \{t'_1, t'_2\}\rangle \in \mathcal{S}$.

Definition 3 (Strong Tamper-Evidence). KESig *is* strongly tamper-evident *if t'_1 and t'_2 are t'_e-safe for all $t'_1, t'_2 > t'_e$.*

Below we consider two weaker versions of tamper-evidence. For both of them, we preserve the power of the adversary. Unlike the strong tamper-evident schemes, both of these weaker versions are allowed to miss some cases of tampering. The first — weaker — guarantees to detect tampering only for simultaneous signatures:

Definition 4 (Perfectly Synchronous Tamper-Evidence). KESig *is* perfectly-synchronous tamper-evident *if* $\{t'_1, t'_2\}$ *are* t'_e*-safe for any* t'_1, t'_2, t'_e*, such that* $t'_1 = t'_2 > t'_e$.

This notion of synchronicity can be relaxed significantly: namely, we can tolerate any distance between the time periods t_1, t_2 of the signatures, as long as they are closer to each other than some factor $(1/\alpha)$ of their distance to the exposure time t_e:

Definition 5 (α-Synchronous Tamper-Evidence). KESig *is* α-synchronous tamper-evident *if* $\{t'_1, t'_2\}$ *are* t'_e*-safe for any* t'_1, t'_2, t'_e *such that* $\min(t'_1, t'_2) - t_e > \alpha|t'_1 - t'_2|$.

Note: 0-synchronous tamper-evidence is equivalent to the strong one of definition 3, while perfectly synchronous tamper-evidence of definition 4 corresponds to ∞-synchronous tamper-evidence. We abbreviate *α-synchronous tamper-evident* as *α-te* (e.g., KESig in definitions 5, 4, and 3 are α-te, ∞-te, and 0-te, respectively).

Possible relaxations. One might wish to further relax the above definitions, e.g., to achieve more efficient constructions. Such relaxations may include limiting adversary powers and/or allowing not self-consistent schemes (i.e., "self-consistent" only with high probability), and/or allowing the probability of missing divergence to be much greater (e.g., less than some constant, say 1%), etc. In this paper we do not consider any such relaxations.

3 Constructions

This section proposes constructions for the strongly and synchronous tamper-evident schemes. The subsequent Section 4 shows that these constructions are optimal (at least up to a constant factor), by proving the matching lower bounds.

Our constructions are generic in the sense that they can be based on any ordinary signature scheme. Also they easily generalize to allow addition of tamper-evidence to other signature models, such as forward-secure or intrusion-resilient.

3.1 Strongly Tamper-Evident Scheme

Construction: Intuitively, this construction extends an ordinary signature by simply appending to it a sequence of $\langle$signature - public key$\rangle$ pairs, where each signature is to be verified using the corresponding public key of the pair. The public keys (and corresponding secret keys) are generated at random, one per time period. So, a tamper-evident signature at time t includes t of these randomly generated (and uncertified) public keys. When the signer's secrets are exposed at time t_e, the adversary learns all the t_e secret keys corresponding to these t_e randomly generated public keys. But after t_e, the signer generates new public-secret key pairs, such that the secret keys for these are *not* known to the adversary. So, the adversary must either use different public keys for $t > t_e$ — which enables detection of divergence, or must forge the signatures for the signer's public keys for periods $t > t_e$, without knowing the corresponding secret keys.

Formally, let Σ be any ordinary signature scheme. $\mathcal{S}_i$ denotes a specific instance of Σ with the corresponding private and public keys $\mathcal{S}_i.SK, \mathcal{S}_i.PK$ generated by the Σ.Gen algorithm.

Define $\mathsf{KESig.Gen}$ to be $\Sigma.\mathsf{Gen} \to (\mathsf{KESig}.SK_0 = \mathcal{S}_0.SK, \mathsf{KESig}.PK = \mathcal{S}_0.PK)$.
$\mathsf{KESig.Upd}(SK_{t-1})$, for $t \le T$, runs $\Sigma.\mathsf{Gen} \to \mathcal{S}_t.\langle PK, SK\rangle$. These keys are appended to the current key $\mathsf{KESig}.SK_{t-1}$ yielding the next period's key $\mathsf{KESig}.SK_t = \langle t; \mathcal{S}_0.SK; \langle \mathcal{S}_i.PK, \mathcal{S}_i.SK\rangle_{i=1 \texttt{ to } t}\rangle$.
$\mathsf{KESig.Sign}(SK_t, t, M)$, for $t \le T$, runs $\mathcal{S}_i.\mathsf{Sign}(SK, \langle t, M\rangle) \to \sigma_i$ for all $i = 0$ $\texttt{to}$ t, generating the signature $\widehat{\sigma}_{t,M} = \langle t, \sigma_0, \langle \mathcal{S}_i.PK, \sigma_i\rangle_{i=1 \texttt{ to } t}\rangle$.
$\mathsf{KESig.Vf}(PK, M, \langle t, \sigma_0, \ldots\rangle)$, returns **valid** if $\mathcal{S}_i.\mathsf{Vf}(PK, \langle t, M\rangle)$ for all $i : 0 \le i \le t$ (i.e., all the ordinary signatures are verified).[6]

Finally, to test for foul play, $\mathsf{KESig.Div}(\widehat{\sigma}_{t_1,M_1}, \widehat{\sigma}'_{t_2,M_2})$ first verifies all the $\mathcal{S}$-signatures (except the very first; their numbers must match the corresponding time periods) in both $\widehat{\sigma}_{t_1,M_1}$ and $\widehat{\sigma}'_{t_2,M_2}$ for $\langle t_1, M_1\rangle$ and $\langle t_2, M_2\rangle$ respectively, and if any are invalid, it returns **foul**.[7] If all the $\mathcal{S}$-signatures are valid, it returns **ok** if one of the sequences $\langle \mathcal{S}_i.PK\rangle_{i=1 \texttt{ to } t_1}$ and $\langle \mathcal{S}'_j.PK\rangle_{j=1 \texttt{ to } t_2}$ from $\widehat{\sigma}_{t_1,M_2}, \widehat{\sigma}_{t_2,M_2}$ respectively, is a prefix of the other. Otherwise, $\mathsf{KESig.Div}$ returns **foul**.

Claim. KESig is as secure as Σ (in the sense of [GMR88]).
Assuming that Σ is secure, KESig is strongly tamper-evident (0-te).

Proof sketch: The proof of signature security is trivial, because our signature simply contains the ordinary signature, appended with random values, which can be easily simulated.

It is also obvious that our scheme is self-consistent.

For the tamper-evidence proof, reduce forging a signature $\mathcal{S}$ to F fooling the Div test. Suppose that we are given a public key $\mathcal{S}.PK$ (and no corresponding secret key). Suppose that we are also given a signature oracle access to the $\mathcal{S}$-signer. The goal is to use forger F — violating the tamper-evidence of our scheme — to generate an non-queried signature valid for $\mathcal{S}.PK$. To achieve this, guess a time period $j(= t_2)$ for which F will fool the Div test, and set $\mathcal{S}_j.PK \leftarrow \mathcal{S}.PK$. All the other parameters and keys are generated by the simulator at random. Then the (adaptive) queries of F can be satisfied by the simulator either directly or with the help of the $\mathcal{S}$-signer oracle.

If F chooses $t_e < j$ and $t_1, t_2 \ge j$, and succeeds in generating $\langle t_2, \sigma_2\rangle$ which passes that $\mathsf{KESig.Div}$ test, then σ_2 contains the $\mathcal{S}$-signature for $\langle t_2, M\rangle$ for some M. If F succeeds, then $\langle t_2, M\rangle$ was never queried, and thus is a forgery for the $\mathcal{S}$. $\square$

Note: inclusion of t in $\langle t, M\rangle$ is needed to prevent the truncation attack. Namely, F obtains from the signature oracle for S the signature $\sigma_1 = \sigma_{\langle t_1, M\rangle}$ for the message M at time t_1. Then, if t is not included in all the messages for any $t_2 < t_1$, $\sigma_2 = \sigma_{\langle t_2, M\rangle}$ is a prefix of σ_1, and thus can be obtained from it by a simple truncation.

3.2 Perfectly Synchronous Tamper-Evident Scheme

Simple construction: Suppose we use the above scheme, but we are guaranteed that $t_1 = t_2 = t$. Then we can actually drop all the keys and signatures $\langle \mathcal{S}_i.PK, \sigma_i\rangle_{i=1 \texttt{ to } t-1}$, leaving only the "real" signature $\mathcal{S}_0$ and the last period's appended signature $\langle \mathcal{S}_t.PK, \sigma_t\rangle$.

The proof above is essentially unaffected by this omission.

The benefit to the efficiency is, of course, dramatic: now a tamper-evident signature is just twice as long as the ordinary one.

[6] It is feasible to have the verification of the uncertified keys as part of Div, but it would require changing the functional definition to pass the message to Div. Also, instead of signing the message with all the keys, it is possible to form a certification chain. This variant can be more efficient in some aspects: the chain does not need to regenerated for each signature. Moreover, the chained construction must be used to achieve the universal indisputability of the tamper evidence discussed in Section 5. However, the version in the main text appears slightly simpler to discuss. In particular, the security proof of the chained version requires random oracles.

[7] This may not be needed, if these signatures have been verified by $\mathsf{KESig.Vf}$, as suggested in the above footnote.

Tree-based construction. A similar synchronous tamper-evident scheme can be obtained using tree-based constructions for forward-secure and intrusion-resilient schemes [BM99,MMM02,Itk02]. Indeed these constructions served as one of the inspirations for this work. While not as efficient as the previous synchronous tamper-evident scheme, the tree-based construction below provides a somewhat more flexible synchronicity restriction than the above scheme as well as some intuition for our subsequent constructions for the α-synchronous schemes.

Intuitively all of these tree-based schemes construct a "certification hierarchy" tree using $\mathcal{S}$-signatures. In this tree, each leaf corresponds to a time period and each node has a public-secret key-pair corresponding to it. The root public key is the public key of the tree-based scheme. And each signature is generated using the corresponding leaf keys and includes the certification path from the leaf public key to the root. Thus each tree-based scheme signature includes a logarithmic number of ordinary signatures (all, but one, are computed at most once per period, independently of the message being signed).

The hierarchy is actually not constructed all at once, but rather generated as needed.[8]

Our main observation here is that, as for the strong tamper-evidence, the $\mathcal{S}$-key-pairs can be generated in a randomized fashion (i.e., without pseudo-randomness, as in, say, [Kra00]). This way, even though a similar (actually even slightly larger) number of keys is generated by S in t time periods, each signature must include only $O(\lg t)$ of these keys, and a signature for each of these $O(\lg t)$ keys. Moreover, all but one (leaf) signatures are computed only once per period (or even less frequently) and are simply re-used for each signature.

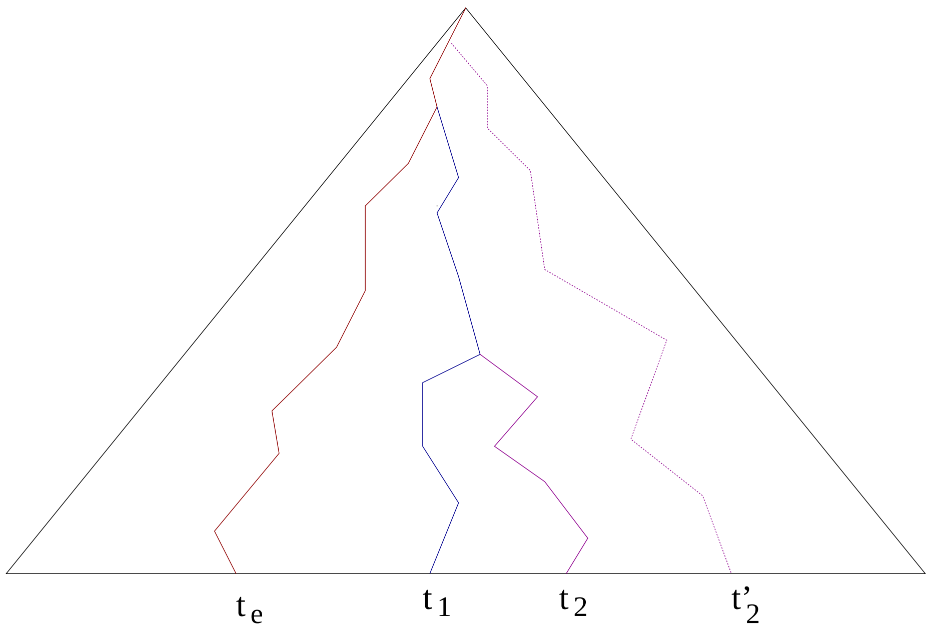

Fig. 2. *Tree-based scheme. Here, it is easy to see that the common prefix of the paths from root to t_1 and t_2 is not a prefix of the path from the root to t_e. On the other hand, t_1 and t_2' do not have this property. Thus,* Div *test can detect divergence for times t_1, t_2 but not t_1, t_2'.*

[8] The original tree-based schemes [BM99,MMM02,Itk02] stored the secret keys of the "right path" for the current leaf (see [Itk02]). For the scheme here, since we are not concerned with forward-security, we can simplify to store the secrets corresponding to the nodes on the path from the current leaf to the root (instead of its right path).

Intuitively, this scheme has somewhat looser synchronicity restrictions than the perfectly synchronous tamper-evidence: it can detect divergence as long as the paths from root to t_1 and t_2 diverge after they diverge with t_e (or in other words, the common prefix of t_1 and t_2 is not a prefix of t_e, see Fig. 2). But it still falls short of the α-te.

Next, we generalize the above tree construction to achieve α-synchronous tamper-evidence for any constant α. As for the tree construction, the cost is only $O(\lg t)$.

3.3 Separating Sets and α-TE Schemes

$\mathcal{C}$-SCHEMES. Let $\mathcal{C}$ be a collection of contiguous sets (intervals) of integers (time period numbers). Define a *$\mathcal{C}$-scheme* as follows: Let a different public/secret key pair $(\mathcal{S}_I.PK, \mathcal{S}_I.SK)$ correspond to each interval $I \in \mathcal{C}$ (the key pair is generated randomly at the beginning of the interval, and is destroyed at the end of it). Each $\mathcal{C}$-signature for the time period t contains a $\mathcal{S}_I$-signature $\langle \mathcal{S}_I.PK, \sigma_I \rangle$ generated using $\mathcal{S}_I.SK$ for each interval $I \in \mathcal{C}$ such that $t \in I$.[9] Let $\mathcal{C}$ contain the infinite interval I_0 of all the integers; $\mathcal{S}_{I_0}.PK$ is the public key of the $\mathcal{C}$-scheme.

Say, that $\mathcal{C}$ is an *α-separating* collection if for any $t_e < t_1, t_2$: $|t_e - \min(t_1, t_2)| > \alpha|t_1 - t_2|$, there exists an interval $I \in \mathcal{C}$ such that $t_1, t_2 \in I$ but $t_e \notin I$.

Lemma 1. *Let $\mathcal{C}$ be an α-separating collection. Then $\mathcal{C}$-scheme is α-synchronous tamper-evident.*

Indeed, since $\mathcal{C}$ is α-separating, there exists I, such that $t_1, t_2 \in I$ but $t_e \notin I$. Thus, both signatures $\widehat{\sigma}_{t_1,M_1}$ and $\widehat{\sigma}_{t_2,M_2}$ must use the same public key $\mathcal{S}_I.PK$. However, $\mathcal{S}_I.SK$ was not created – and thus was not known – at the time of the latest exposure t_e. Thus F cannot generate the $\mathcal{S}_I$-signature for $\mathcal{S}_I.PK$ (i.e., we could reduce forging $\mathcal{S}$ signatures to F' success). □

The 0-, α- and ∞- te schemes of the above sections can be viewed as such $\mathcal{C}$-schemes: For the 0-te scheme we used $\mathcal{C}_0 = \{\{t, t+1, \ldots\}\ \mathit{for\ all}\ t\}$. Our ∞-te scheme used $\mathcal{C}_\infty = \{\{t\}\ \mathit{for\ all}\ t\}$. The tree-based schemes use $\mathcal{C}_{tree} = \{\{i2^j, \ldots, (i+1)2^j - 1\}\ \mathit{for\ all}\ j \geq 0,\ i > 0\}$.[10]

Thus, constructing α-te schemes is reduced to a combinatorial problem of constructing α-separating collections; the number of intervals containing t, for each t, corresponds to the scheme's cost.

Constructing α-Separating Collections

Fact 1 *For any $t_1, t_2 : |t_1 - t_2| = d$, any interval of size $\leq d(1+\alpha)$ containing t_1, t_2 cannot also contain such t_e that $|t_e - \min(t_1, t_2)| > \alpha|t_1 - t_2|$.*

Indeed, $|t_e - \min(t_1, t_2)| > \alpha|t_1 - t_2| = \alpha d \Rightarrow |t_e - \max(t_1, t_2)| > (1+\alpha)d$.

Let β be any constant such that $0 < \beta < \alpha$. Intuitively, we define intervals as in Fig. 3: the intervals of length $d(1+\alpha)$ are going to be shifted by multiples of $d(\alpha - \beta)$. Then, for any t_1, t_2 such that $d \leq |t_1 - t_2| \leq d(1+\beta)$ there exists an interval (i) containing both t_1 and t_2 and (ii) not containing t_e satisfying the α-synchronicity: $|t_e - \min(t_1, t_2)| > \alpha|t_1 - t_2|$.

For t_1, t_2 such that $|t_1 - t_2| > d(1+\beta)$ intervals of size $> d(1+\beta)(1+\alpha)$ are needed. In other words the interval lengths can increase by a factor of $1+\beta$.

[9] Thus, the signer must have the same secret key $\mathcal{S}_I.SK$ for all $t \in I$. But then if the signer has $\mathcal{S}_I.SK$ during periods $t_1 < t_2$, then this key must also be in the signer's possession during all periods $t : t_1 \leq t \leq t_2$. Therefore, requiring that the sets in $\mathcal{C}$ be contiguous reflects the security requirements in a natural way.

[10] The version of Sec. 3.2 actually allowed $i = 0$; though using it with a balanced hierarchy also bounds t. If t is unbounded, then allowing $i = 0$ leads to each t belonging to infinite number of intervals and thus to an infinite number of keys for each time t. The $\mathcal{C}_{tree}$ above eliminates all intervals containing 0: these cannot help separation of any t_1, t_2 from t_e. This results in each t belonging to only $1 + \lg t$ intervals.

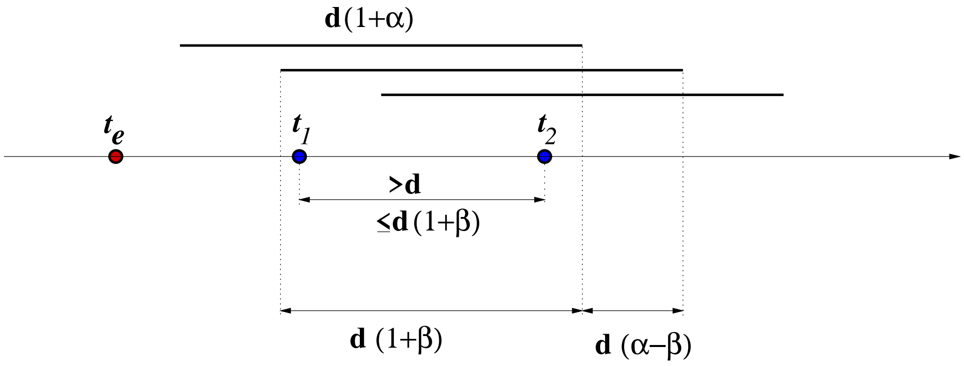

Fig. 3. Arranging intervals of size $d(1+\alpha)$. Any two points at distance $\leq d(1+\beta)$ will be contained in one of the intervals. At the same time, for any t_1, t_2, separated by distance $> d$, the interval containing t_1, t_2 cannot contain such a t_e which would satisfy α-synchronicity requirement: $|t_e - \min(t_1, t_2)| > \alpha|t_1 - t_2|$. Thus, these intervals "take care" of the signatures separated by distances $> d$ and $\leq d(1+\beta)$

More formally, for *level* j, let $d_j \stackrel{def}{=} (1+\beta)^j$. Then the corresponding *interval length* is $L_j \stackrel{def}{=} d_j(1+\alpha)$, separating $t_1, t_2 : d_j \leq |t_1 - t_2| \leq d_j(1+\beta)$ from t_e. The *shift* $\Delta_j \stackrel{def}{=} d_j(\alpha - \beta)$. The intervals for level are shifted by Δ_j, guaranteeing that each pair t_1, t_2 as above belongs to some interval of the level. Define interval $I_{\alpha,\beta,i,j} = [i\Delta_j, \ldots, i\Delta_j + L_j - 1]$. Note: $|I_{\alpha,\beta,i,j}| = L_j$. We refer to j as the level of $I_{\alpha,\beta,i,j}$, and i as its displacement.

Define $\mathcal{C}_{\alpha,\beta} = \{\{t\} : t > 0\} \bigcup \{I_{\alpha,\beta,i,j} : i > 0,\ j \geq 0\}$. Intuitively, the singleton sets deal with the cases $|t_1 - t_2| = 0$ (i.e., $t_1 = t_2$), while the level j intervals handle the distances $|t_1 - t_2|$ such that $d_j < |t_1 - t_2| \leq d_{j+1}$.

In $\mathcal{C}_{\alpha,\beta}$, no more than $L_j/\Delta_j + 1 = (1+\alpha)/(\alpha - \beta) + 1$ intervals of any one level contain each point t. Moreover, t cannot be contained by any intervals of level j such that $t < \Delta_j = (1+\beta)^j(\alpha - \beta)$. Thus, t can be contained by intervals of at most $\log_{1+\beta}(t/(\alpha - \beta))$ levels.

So, the total number of intervals containing time period t is upper-bounded by $1 + ((1+\alpha)/(\alpha - \beta) + 1) \cdot \log_{1+\beta} t/(\alpha - \beta)$. Using an $\mathcal{C}_{\alpha,\beta}$-scheme to achieve α-synchronous tamper-evidence yields the same upper-bound (plus one for the I_0) on the number of keys used for and ordinary signatures included in an α-te signature at time t. For constants $\alpha > \beta > 0$, the above formula simplifies to $O(\lg t)$. We leave out of this version of the paper the question of computing the values of β for the given α, which would yield the best constant factors hidden by the big-O notation. It may be helpful to consider the case of 2-synchronicity: using $\beta = 1$, the number of keys stored and used at time t is at most $2 + 4\lg t$. Thus we have the following lemma:

Lemma 2. *For any constant $\alpha > 0$ and ordinary signature scheme Σ, there exists an α-synchronous tamper-evident scheme* KESig*, storing $O(\lg t)$ keys and generating $O(\lg t)$ Σ-signatures for each* KESig*-signature at time t.*

3.4 Lower Bounds For The Subset Separation Schemes

In this section we consider only tamper-evident schemes based on using an ordinary signature scheme to separate S from F by requiring that for any $t_e < t_1, t_2$ there exists a signature scheme instance with a public key p (using corresponding secret key s) such that the signatures for times t_1, t_2 must both use this instance, but at time t_e the key s was not known to the signer (and thus to the attacker, who exposed all the secrets of the signer at times $\leq t_e$). Such schemes are equivalent to the subset separation schemes. We provide the lower-bounds for this restricted class. The general case lower bounds proven in the subsequent section subsume those of this section. However, we leave them here due to their more intuitive clarity.

Claim. Any ste scheme requires at least $t-2$ ordinary keys/signatures to be used for each ste signature at time t.[11]

Indeed, consider the signature at time $t_2 = t$. Let $t_1 = t-1$, and $t_e = t-2$. Then there must be a key shared by t_1, and t_2, but not t_e. Now, let $t_1 = t-2$, $t_e = t-3$. Then, there must be a different key shared by $t, t-2$, but not $t-3$. Let's make one more step before we generalize: let $t_1 = t-3$, $t_e = t-4$. Then, t and $t-3$ must share a key not shared by t_e. This key must clearly be different from the previous. But it also must be different from the one shared by t and $t-1$ but not $t-2$: because if the key is known at times $t-3$ and t, then it must be known also at $t-2$. Thus, to generalize: for each $1 < i < t-1$, there must be a different key shared by $t_1 = t$ and $t_2 = t-i$ but not $t_e = t-i-1$. □

The above claim proves that our scheme (Sec. 3.1) is optimal within this general approach.

That claim generalizes to α-synchronicity:

Claim. For any constant α, any α-synchronous scheme for each signature at time t must generate $\Omega(\lg t)$ ordinary signatures (using as many different keys).

Indeed, let $\epsilon > 0$ be some arbitrarily small constant, set $t_2 = t$, and initially let $t_1 = t-1$ and $t_e = \lfloor t-(1+\alpha)-\epsilon \rfloor$. There must be a key that separates t_1, t_2 from t_e. And it must be different from the one that separates $t_2 = t$ and $t_1 = \lfloor t-(1+\alpha)-\epsilon \rfloor$ from $t_e = \lfloor t-(1+\alpha)-(1+\alpha)^2-2\epsilon \rfloor$. This step can be iterated k times as long as $t \geq \sum_{i=0}^{k}(1+\alpha)^i = (1+\alpha)^{k+1}/\alpha$. Thus, at least $\Omega(\lg t)$ ordinary signatures (all using different keys) must be generated for each TE signature at time t. □

The next section extends these lower bounds to the most general tamper-evident schemes.

4 General Lower Bounds

Let KESig be some key-evolving signature scheme with a divergence test, according to the definitions in Sec. 2.1, and the adversary as in the Definition 1.

Define *support of* t, $\mathbf{supp}_{\mathsf{KESig}}(t)$, to be a longest increasing chain $t_0, t_1, \ldots, t_l = t$, such that t, t_{i+1} are t_i-safe in the given KESig scheme for all $i : 0 \leq i \leq l-1$ (thus, $t_j, t_{j'}$ are also t_i-safe for any $j, j' > i$). *Order of* t, $\mathbf{ord}_{\mathsf{KESig}}(t)$, is defined to be the length l of this chain (measured in the number of possible values for the exposure time period t_i). For example, if KESig is ste, then for any t, $\mathbf{ord}_{\mathsf{KESig}}(t) = t+1$, since we can set $t_i = i-1$ for all $0 \geq i \geq t+1$; exposure time $t_e = -1$ corresponds to having no exposure. For α-te KESig, $\mathbf{ord}_{\mathsf{KESig}}(t) = \Theta(\log_{1+\alpha} t)$.

Recall that k is the security parameter used in the definition of safety (Def. 2).

We now show that the length of the signature at time t must be at least $\mathbf{ord}_{\mathsf{KESig}}(t) \cdot k$.

Let $t_0, t_1, \ldots, t_l = t$ be a support of t. Let F' be any (forger) algorithm; unless stated otherwise, we do *not* assume that F' has any access to the legitimate signer's secrets or even signatures. In this respect, F' is significantly weaker than the adversary F of the Definition 1. Generate an instance of KESig, and let the legitimate signer S generate signatures $\langle t_i, \sigma_i \rangle$ for some message m and all $i = 1, \ldots, l$ (signature for t_0 is not needed, since t_0 is used only as a possible exposure time period; often $t_0 = -1$). For a signature $\langle t, \sigma \rangle$ of some other message m' ($\neq m$) at time t, define event $C_i(\langle t, \sigma \rangle)$ to be $\mathsf{KESig.Div}(\langle t_i, \sigma_i \rangle, \langle t, \sigma \rangle) = \mathbf{ok}$. Let $C_{[j]}(\langle t, \sigma \rangle)$ be the conjunction of all $C_{i=1,\ldots,j}(\langle t, \sigma \rangle)$.

[11] This claim, including the proof, was suggested by Leonid Levin.

Lemma 3. $Prob[\, F' \to (t,\sigma): \ s.t.\ C_{[l]}(\langle t,\sigma\rangle)\,] < 1/2^{kl}$

Proof: Let $P_j \stackrel{def}{=} Prob[\, F' \to (t,\sigma): \ \text{s.t. } C_{[j]}(\langle t,\sigma\rangle)\,]$, for $0<j\le l$, and (vacuously) set $P_0=1$. Then the lemma states that $P_l < 1/2^{kl}$.
Let $F' \to (t,\sigma)$. Then, $P_j = Prob[C_j(\langle t,\sigma\rangle) \,|\, C_{[j-1]}(\langle t,\sigma\rangle)] \cdot Prob[C_{[j-1]}(\langle t,\sigma\rangle)]$. Substitute $P_{j-1} = Prob[C_{[j-1]}(\langle t,\sigma\rangle)]$ into the above to get $P_j = Prob[C_j(\langle t,\sigma\rangle)|C_{[j-1]}(\langle t,\sigma\rangle)] \cdot P_{j-1}$.
Let $S(t_i)$ be the full record of the legitimate signer's evolution up to and including time period t_i (that is the record of all the signer's information up until that time, including the secret keys).
Then, $Prob[C_j(\langle t,\sigma\rangle)|C_{[j-1]}(\langle t,\sigma\rangle)] \le Prob[F'^{S(t_{j-1})} \to (t,\sigma') : \mathsf{KESig.Div}(\langle t_j,\sigma_j\rangle, \langle t,\sigma'\rangle) = \mathbf{ok}] \le Prob[F^S \to \langle t'_e = t_{j-1}, \{(t'_1 = t_j, \sigma'_1 = \sigma_j), (t'_2 = t, \sigma'_2)\}\rangle : \mathsf{KESig.Div}(\langle t'_1,\sigma'_1\rangle, \langle t'_2,\sigma'_2\rangle) = \mathbf{ok}] < 1/2^k$, where F is the forger from Definitions 1, 2.

Putting it all together we get $P_j < P_{j-1} \cdot 1/2^k$. Thus, $P_j < 1/2^{kj}$. □

Let $C_l(\langle t,\sigma\rangle)$ be true (e.g., $\langle t,\sigma\rangle$ is generated by the legitimate signer S). If $|\sigma| < lk$ then a random $\sigma' = \sigma$ with probability $> 1/2^{lk}$, which contradicts Lemma 3. Thus, the following theorem follows as a corollary from the Lemma:

Theorem 1. *Let* $\mathsf{KESig.Sign} \to \langle t,\sigma\rangle$ *for some message. Then* $|\sigma| > k \cdot \mathbf{ord}_{\mathsf{KESig}}(t)$.

Since for the strongly tamper evident schemes $\mathbf{ord}_{\mathsf{KESig}}(t) = t+1$, and for the α-synchronous schemes (for finite $\alpha>0$) $\mathbf{ord}_{\mathsf{KESig}}(t) = \Theta(\lg t)$, we get the following corollaries:

Corollary 1 (Strong Tamper-Evident Signature Length).
$|\sigma| > k \cdot (t+1)$ *for any* ste $\mathsf{KESig.Sign} \to \langle t,\sigma\rangle$.

Corollary 2 (α-Synchronous Tamper-Evident Signature Length).
$|\sigma| = \Omega(k \cdot \lg t)$ *for any* α-te $\mathsf{KESig.Sign} \to \langle t,\sigma\rangle$.

5 Discussion

UNIVERSAL EVIDENCE. We can modify our constructions (e.g., using chaining as suggested in the footnote 6) so that any pair of signatures $\langle t_1,\sigma_1\rangle, \langle t_2,\sigma_2\rangle$, which are both valid for the same public key but $\mathsf{Div}(\langle t_1,\sigma_1\rangle, \langle t_2,\sigma_2\rangle) = \mathbf{foul}$, represents a universal and indisputable evidence that either the key has been exposed or that the signer is faking the key exposure.

PKI IMPLICATIONS. Revocation is a traditional method of dealing with the compromised keys. Whatever is the revocation mechanism, the key compromise must be detected first, and then the revocation process followed appropriately. In all the traditional Public Key Infrastructures (PKIs), some party — call it Revocation Authority (RA) — must be convinced that the key is indeed compromised, before it actuates the revocation: typically, generating a revocation note[12].

Whether RA is the signer himself or a CA (or other), convincing RA of the key compromise using the previously existing methods is potentially cumbersome both logistically and legally.

In contrast, our schemes allow *anyone* to detect tampering and present a universally convincing proof of the compromise: two valid but inconsistent signatures as above. In fact, such a proof may serve as a revocation note. Moreover, the legitimate signer S can post on some publicly accessible site his signature for each day (the signature is verified before being posted to avoid denial of

[12] This can be a self-signed "suicide note" (as in PGP or other approaches e.g., [Riv98]); in these cases RA is the signer himself. Alternatively, in the more common PKIs, the revocation note is a part of a Certificate Revocation List (CRL), or a similar data structure, which is generated and certified by some (trusted?) third party, such as the Certification Authority (CA).

service attack). This can serve as an automatic revocation check: instead of checking a CRL, the verifiers can use that signature with the Div algorithm to verify that S has not been compromised. Of course, it is possible that immediately following an exposure, F manages to post her version of the daily signature. But then S will be able to detect the tampering and resolve the conflict by out-of-band means. Alternatively, the server can allow more than one signature to be posted for each user — if two of the posted signatures trigger the tamper-detection test Div, then the other users of the system will also know that the corresponding secret key has been compromised. A potential key-recovery mechanism would then be to allow the server to obtain S's signature over some secure connection, and remove the previous signatures from the server. Then the verifiers will again be able to distinguish S's signatures from the forger's.

The public posting site can be replaced with a more "personal" version: in the case of regular transactions between the signer and a recipient, the recipient can keep the latest signer's signature as a "cookie". Then even after the exposure of the signer's secrets, the adversary cannot impersonate the singer to the recipient (again, except immediately after the exposure).

Symmetric signatures and peer-to-peer setting. Since our constructions are generic, it is possible to use symmetric signatures, and apply the tamper-evidence to the peer-to-peer setting. The use of symmetric signatures only, however, requires the coordinated randomized key evolution between the pair of connected nodes. This can be achieved under the condition that the adversary cannot access some of the information exchanged by the legitimate parties. While this assumes a weaker adversary than the one tolerated with the asymmetric signatures, this model can still be practical in some situations and has the advantage of efficiency offered by the symmetric signatures.

Combining tamper evidence with other features. The tamper-evidence can be combined with other security improvements for signatures: e.g., our constructions can be easily generalized to the forward-secure [BM99] or intrusion-resilient models [IR02,Itk02].

Monotone Signatures. The monotone signatures were defined by Naccache, Pointcheval and Tymen in [NPT01]. These signatures allow updating the verification algorithm. Then, the legitimate signer can reveal some secrets under duress. This would allow the extortionist to generate signatures valid under the current verification algorithm. However, when the signer is released, he can update the verification algorithm in such a way that all the signatures generated by the legitimate signer remain valid under the new verification procedure as well; but the signatures generated by the extortionist are no longer valid.

In contrast to tamper-evident or forward-secure signatures, the definitions of [NPT01] do not include the possibility of also updating the signer keys. We propose to use key-evolving monotone signatures instead. Then we can achieve the main features of the monotone signatures directly from the tamper-evident signatures: The verification algorithm would include checking for tampering using Div and a signature for time period $t-1$. The signer would maintain a "correct" version of the secret key, as well as a version "diverged" in the current period. This divergence cannot be detected against the signature of the "pre-diversion" period $t-1$, used in the current verification. Thus, the signer can release that diverged version under duress. Afterwards, he can update the verification by including a signature for the period t. The attacker can be prevented from generating earlier signatures by combining the above with the forward-security. This method is certainly not any more efficient than the original constructions of [NPT01]; it is given here solely to illustrate the connection of the two concepts. However, the efficiency of the monotone signatures can be improved by the key-evolution: this approach is further developed in [Itk03].

Other Potential Approaches. Suppose, we allow the Div test to occasionally miss divergence and potentially give a false positive: returning **foul** on two legitimate signatures. In addition, assume that F cannot obtain any signatures from S after t_e. Then we may attempt the following approach. Define some metric on the space of public keys, and restrict the distance between consecutive public keys $\mathcal{S}_i.PK, \mathcal{S}_{i+1}.PK$, so that there is still a multiple choice for the next public key. Then evolution of the signer corresponds to a random walk. We can now try to utilize the property that within one random walk, the distance between the positions at times t_1, t_2 is likely to be noticeably smaller than the positions corresponding to t_1 and t_2 on two different random walks (which diverged at some previous time t_e). It is unlikely, however, that this approach will improve on our results above.

Other possible directions for future research include considering more interactive models of authentication (e.g., zero-knowledge proofs of identity [FS86,FFS88]), and extending Div to use more than two signatures to detect tampering (such an extension may impact some of the PKI-related issues discussed in this section)

6 Acknowledgments

The author is very grateful to Leonid Levin and Drue Coles for useful discussions. In particular, discussions with Drue were instrumental in crystallizing the key concepts, while Leonid pointed out the lower bound for the subset separation strongly tamper-evidence schemes. The author is also grateful to the anonymous referees for their comments, and in particular, for pointing the connection with the monotone signatures. Finally, thanks to Drue and Peng Xie for their very helpful comments on the preliminary drafts of the paper.

References

[And97] Ross Anderson. Invited lecture. Fourth Annual Conference on Computer and Communications Security, ACM (see http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf), 1997.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, 3–7 December 2000. Springer-Verlag. Full version available from the Cryptology ePrint Archive, record 2000/002, `http://eprint.iacr.org/`.

[BF97] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. 17th International Advances in Cryptology Conference – CRYPTO '97*, pages 425–439, 1997.

[BFN98] M. Blaze, J. Feigenbaum, and M. Naor. A formal treatment of remotely keyed encryption. In *Advances in Cryptology – EUROCRYPT '98*, pages 251–265, 1998.

[Bla95] Matt Blaze. High-bandwidth encryption with low-bandwidth smartcards. Technical report, AT&T Bell Laboratories, October 1995. ftp://research.att.com/dist/mab/card_cipher.ps. Draft.

[Bla96] Matt Blaze. High-bandwidth encryption with low-bandwidth smartcards. In Dieter Grollman, editor, *Fast Software Encryption: Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 33–40, Cambridge, UK, 21–23 February 1996. Springer-Verlag.

[BM99] Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 15–19 August 1999. Revised version is available from `http://www.cs.ucsd.edu/~mihir/`.

[CDH+00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469. Springer-Verlag, 14–18 May 2000.

[CHH00] Ran Canetti, Shai Halevi, and Amir Herzberg. Maintaining authenticated communication in the presence of break-ins. *Journal of Cryptology*, 13(1):61–105, January 2000.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 1990, 20–24 August 1989.

[DKXY] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. Unpublished Manuscript.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Knudsen [Knu02].

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987, 11–15 August 1986.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *Fourth ACM Conference on Computer and Communication Security*, pages 100–110. ACM, April 1–4 1997.

[HJJY00] Johan Håstad, Jakob Jonsson, Ari Juels, and Moti Yung. Funkspiel schemes: an alternative to conventional tamper resistance. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 125–133. ACM Press, 2000.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer-Verlag, 19–23 August 2001.

[IR02] Gene Itkis and Leonid Reyzin. Intrusion-resilient signatures, or towards obsoletion of certificate revocation. In Moti Yung, editor, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 August 2002. Available from `http://eprint.iacr.org/2002/054/`.

[Itk02] Gene Itkis. Intrusion-resilient signatures: Generic constructions, or defeating strong adversary with minimal assumptions. In *Third Conference on Security in Communication Networks (SCN'02)* [SCN02].

[Itk03] Gene Itkis. Key-evolving monotone signatures. draft, 2003.

[Knu02] Lars Knudsen, editor. *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science. Springer-Verlag, 28 April–2 May 2002.

[KR02] Anton Kozlov and Leonid Reyzin. Forward-secure signatures with fast key update. In *Third Conference on Security in Communication Networks (SCN'02)* [SCN02].

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *Seventh ACM Conference on Computer and Communication Security*. ACM, November 1–4 2000.

[Luc97] Stefan Lucks. On the security of remotely keyed encryption. In Eli Biham, editor, *Fast Software Encryption: 4th International Workshop*, volume 1267 of *Lecture Notes in Computer Science*, pages 219–229, Haifa, Israel, 20–22 January 1997. Springer-Verlag.

[MMM02] Tal Malkin, Daniele Micciancio, and Sara Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Knudsen [Knu02].

[NPT01] David Naccache, David Pointcheval, and Christophe Tymen. Monotone signatures. In P. Syverson, editor, *Financial Cryptography*, volume 2339 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 2001.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *10-th Annual ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer-Verlag, 8–11 April 1991.

[PP97] Torben Pryds Pedersen and Birgit Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, 26(2):291–330, 1997.

[Riv98] Ronald L. Rivest. Can we eliminate certificate revocation lists? In Rafael Hirschfeld, editor, *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[SCN02] *Third Conference on Security in Communication Networks (SCN'02)*, Lecture Notes in Computer Science. Springer-Verlag, September 12–13 2002.