A Forward-Secure Public-Key Encryption Scheme*

Jonathan Katz[†]

September 5, 2002

Abstract

Cryptographic computations are often carried out on insecure devices for which the threat of key exposure represents a serious and realistic concern. In an effort to mitigate the damage caused by exposure of secret data stored on such devices, the paradigm of *forward security* was introduced. In this model, secret keys are updated at regular intervals throughout the lifetime of the system; furthermore, exposure of a secret key corresponding to a given interval does not enable an adversary to "break" the system (in the appropriate sense) for any *prior* time period. A number of constructions of forward-secure digital signature schemes and symmetric-key schemes are known.

We present the first construction of a forward-secure public-key encryption scheme whose security is based on the bilinear Diffie-Hellman assumption in the random oracle model. Our scheme can be extended to achieve chosen-ciphertext security at minimal additional cost. The construction we give is quite efficient: all parameters of the scheme grow (at most) poly-logarithmically with the total number of time periods.

1 Introduction

Exposure of secret keys can be a devastating attack on a cryptosystem since such an attack typically implies that all security guarantees are lost. Indeed, standard notions of security offer no protection whatsoever once the secret key of the system has been compromised. With the threat of key exposure becoming more acute as cryptographic computations are performed more frequently on small, unprotected, and easily-stolen devices (e.g., mobile phones), new techniques are needed to deal with this concern.

A number of methods have been introduced in an attempt to counter this threat. Of course, one may try to eliminate the occurrence of key exposure entirely by using tamper-proof hardware or some variation of this idea. While this is an important direction of research, such techniques are often too expensive or otherwise not practical for the intended application. A second class of approaches assumes instead that key exposure will occur and seeks to minimize the resulting damage. Secret sharing [21], threshold cryptography [9], and proactive cryptography [20, 14] — in which secrets are "split" across multiple devices — are perhaps the best-known approaches in this vein. Unfortunately, such solutions tend to

*This work has recently been superseded by [8].

[†]jkatz@cs.umd.edu. Department of Computer Science, University of Maryland (College Park). Portions of this work were done while at Columbia University.

be costly; for one thing, they require multiple devices where a single device would have previously sufficed. Furthermore, threshold computations are inherently interactive and hence require extensive coordination between different hosts; in settings where communication is at a premium (e.g., wireless networks), such computations become prohibitive. Finally, note that for devices which are insecure and are thus expected to have a high incidence of physical compromise, exposing secrets stored on multiple devices may not be significantly more difficult than exposing the secrets stored on a single device.

In an effort to address these concerns, the notion of *forward security* was recently proposed by Anderson [3] and later formalized by Bellare and Miner [4]. The basic idea is to divide the lifetime of the system into N intervals (or time periods) labeled $0, \ldots, N-1$. The device begins by storing secret key SK_0; this secret key will "evolve" with time so that SK_0 will be used during period 0, SK_1 will be used during period 1, and so on. At the beginning of time period i the device applies some function to the "previous" key SK_{i-1} in order to derive the "current" key SK_i; key SK_{i-1} is then deleted. The current key is used for all (secret) cryptographic operations during the corresponding period. The public key (assuming one exists) is *never* updated; instead, it remains fixed throughout the lifetime of the system. A forward-secure scheme guarantees that an adversary who learns SK_i for some i will be unable to "break" the security of the system (in the appropriate sense) for all time periods *prior* to i. Note that since the adversary obtains all secrets existing at that point in time, the model inherently cannot prevent the adversary from "breaking" the security of the system at time periods subsequent to i.

The forward-secure paradigm is quite general and can be applied to essentially any cryptographic primitive; most research thus far, however, has focused on the construction of forward-secure signature schemes and the related case of identification schemes. The generic forward-secure signature scheme of Anderson [3] was improved by Bellare and Miner [4] who also present the first efficient and non-generic construction. These initial works inspired a sequence of improved constructions yielding more efficient forward-secure signature schemes as well as schemes giving tradeoffs among the various parameters [18, 2, 16, 19]. Integrating forward-secure signature schemes and threshold techniques has also been investigated [1, 22]. The only prior instance in which forward-security was considered for primitives other than signature/identification schemes is the work of Bellare and Yee [5] focusing on the private-key setting.

Motivated by work on forward security, the related notion of key-insulated cryptography has recently been introduced [10, 11]. Although in both models the stored secret keys are updated so as to limit the effect of key exposures, the models are in fact incomparable. On one hand, the key-insulated model achieves a stronger level of security in that, even after *multiple* key exposures occur, *all* non-exposed time periods remain secure. On the other hand, the price for this additional security is the (necessary) assumption of a (semi-)trusted server which is never compromised and with which the device must interact at the beginning of each time period. In certain scenarios an assumption of this form is clearly unwarranted.

1.1 Our Contribution

As mentioned above, almost all previous research on forward-secure primitives (with the exception of [5]) has focused on the case of digital signature schemes. Indeed, the question of

Key generation time	$\mathcal{O}(\log N)$
Encryption/Decryption time	$\mathcal{O}(\log N)$
Key update time	$\mathcal{O}(\log N)$
Ciphertext length	$\mathcal{O}(\log N)$
Public key size	$\mathcal{O}(1)$
Secret key size	$\mathcal{O}(\log^2 N)$

Table 1: Summary of dependencies on the total number of time periods N.

whether a (non-trivial) forward-secure public-key encryption scheme could be constructed has been open since the introduction of the forward-security paradigm [3, 4]. We show here a simple and efficient construction of a forward-secure public-key encryption scheme whose security may be based on the bilinear Diffie-Hellman assumption (cf. [6]) in the random oracle model. Using the Fujisaki-Okamoto transformation [12], chosen-ciphertext security can be achieved with minimal additional cost. The dependency of the parameters of our scheme on the total number of time periods N is quite good; in all cases, it is at most poly-logarithmic. The key details are summarized in Table 1. We stress that these results should not be interpreted as indicating "merely" asymptotic efficiency; in general (except for the size of secret key storage), our scheme is as efficient as $\log_2 N$ invocations of the Boneh-Franklin identity-based encryption scheme [6] and is therefore quite practical for reasonable values of N.

2 Definitions and Preliminaries

Here, we provide definitions for key-evolving public-key encryption schemes and also define what it means for an encryption scheme to be forward-secure. The former definition is a straightforward adaptation of [4]; the latter, however, is new and requires some care.

Definition 1 A *key-evolving public-key encryption scheme* $\mathsf{ke\text{-}PKE} = (\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ is a 4-tuple of algorithms such that:

- The probabilistic *key generation algorithm* Gen takes as input a security parameter 1^k and the total number of time periods N. It returns a public key PK and an initial secret key SK_0.
- The probabilistic *key update algorithm* Upd takes as input a secret key SK_{i-1} as well as the index i of the current time period. It returns a secret key SK_i for period i.
- The probabilistic *encryption algorithm* Enc takes as input a public key PK, the index i of the current time period, and a message M. It returns a ciphertext C for period i. We always represent the output as a pair $\langle i, C\rangle$ and write $\langle i, C\rangle \leftarrow \mathsf{Enc}_{PK}(i, M)$.
- The deterministic *decryption algorithm* Dec takes as input a secret key SK_i and a ciphertext $\langle i, C\rangle$. It returns a message M. We denote this by $M := \mathsf{Dec}_{SK_i}(\langle i, C\rangle)$.

We require the standard correctness condition; namely, for any (PK, SK_0) output by Gen, any secret key SK_i correctly generated for period i, all M, and all $\langle i, C\rangle$ output by $\mathsf{Enc}_{PK}(i, M)$ we have $M = \mathsf{Dec}_{SK_i}(\langle i, C\rangle)$. ■

A definition for forward-secure public-key encryption (PKE) does not follow immediately from the corresponding definition for signature schemes. For one thing, in a forgery attack the forged signature/message pair is the final output of the adversary whereas in the case of PKE the adversary might be able to perform additional useful "attacks" even after observing a ciphertext $\langle i, C\rangle$. To be more specific, one can imagine two plausible definitions of forward-secure PKE: (1) the adversary must obtain some secret key SK_i *before* requesting ciphertext $\langle j, C\rangle$ for some $j < i$; or (2) the adversary must request ciphertext $\langle j, C\rangle$ *before* he can obtain some secret key SK_i with $i > j$. To make matters worse, neither definition of security seems to imply the other. Our definition captures the strongest notion of security by giving the adversary control over the ordering of the above events.

Definition 2 A key-evolving public-key encryption scheme ke-PKE is *forward-secure in the sense of indistinguishability* (fs-IND) if no PPT adversary has non-negligible advantage in the following game:

Setup: $\mathsf{Gen}(1^k, N)$ is run, yielding output (PK, SK_0). The adversary is given PK.

Attack: The adversary issues one $\mathsf{breakin}(i)$ query and one $\mathsf{challenge}(j, M_0, M_1)$ query, in either order, subject to $0 \le j < i < N$. These queries are answered as follows:

- On query $\mathsf{breakin}(i)$, key SK_i is computed via $\mathsf{Upd}(\cdots \mathsf{Upd}(SK_0, 1), \cdots i)$. This key is then given to the adversary.
- On query $\mathsf{challenge}(j, M_0, M_1)$ a random bit b is selected and the adversary is given $\mathsf{Enc}_{PK}(j, M_b)$.

Guess: The adversary outputs a guess $b' \in \{0, 1\}$. The adversary *succeeds* if $b' = b$.

The advantage of the adversary is defined as the absolute value of the difference between its success probability and $1/2$. ■

Remark 1. When schemes are defined in the random oracle model we additionally allow the adversary to make a polynomially-bounded number of queries to any random oracles used in constructing the scheme. These queries may be interleaved in any order with the breakin and challenge queries.

Remark 2. Following [6, 15, 13], we may define *forward security in the sense of one-wayness* (fs-OWE) in the obvious way. Similarly, we may also define *forward security under chosen-ciphertext attack* (fs-CCA) as the natural extension of Definition 2. We defer the details until the final version of this paper.

2.1 Cryptographic Assumptions

The security of our forward-secure encryption scheme is based on the difficulty of the bilinear Diffie-Hellman (BDH) problem as recently formalized by Boneh and Franklin [6]. This assumption, or variants thereof, has been used for a number of different cryptographic constructions (e.g., [17, 7, 23, 15, 13]). We review the relevant definitions as they appear in [6, 13]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q, where $\mathbb{G}_1$ is represented additively and $\mathbb{G}_2$ is represented multiplicatively. We use a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ for which the following hold:

1. The map $\hat{e}$ is *bilinear*; that is, for all $P_0, P_1 \in \mathbb{G}_1$ and all $x, y \in \mathbb{Z}_q$ we have $\hat{e}(xP_0, yP_1) = \hat{e}(P_0, P_1)^{xy}$.

2. There is an efficient algorithm to compute $\hat{e}(P_0, P_1)$ for any $P_0, P_1 \in \mathbb{G}_1$.

A *BDH parameter generator* $\mathcal{IG}$ is a randomized algorithm that takes a security parameter 1^k, runs in polynomial time, and outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and a map $\hat{e}$ satisfying the above conditions. We define the *BDH problem with respect to* $\mathcal{IG}$ as the following: given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ output by $\mathcal{IG}$ along with random $P, aP, bP, cP \in \mathbb{G}_1$, compute $\hat{e}(P,P)^{abc}$. We say that $\mathcal{IG}$ *satisfies the BDH assumption* if the following is negligible (in k) for all PPT algorithms A:

$$\Pr[(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); P \leftarrow \mathbb{G}_1; a, b, c \leftarrow \mathbb{Z}_q : A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc}].$$

We note that BDH parameter generators for which the BDH assumption is believed to hold can be constructed from Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [6] for details.

3 Forward-Secure Public-Key Encryption

3.1 Overview

We first provide an overview of our construction. Assume for simplicity that the total number of time periods N is a power of 2; that is, $N = 2^t$. We imagine a full binary tree of height t in which the root is labeled with ε (representing the empty string) and furthermore if a node at depth less than t is labeled with w then its left child is labeled with $w0$ and its right child is labeled with $w1$. Let $\langle i \rangle$ denote the t-bit representation of integer i (where $0 \le i \le 2^t - 1$). The leaves of the tree (which are labeled with strings of length t) correspond to successive time periods in the obvious way; i.e., time period i is associated with the leaf labeled by $\langle i \rangle$.

For simplicity, we refer to the node labeled by w as simply "node w". Every node w in the tree will have an associated secret key sk_w; recall further that there is one public key PK which remains fixed throughout the lifetime of the scheme. The properties of our PKE construction can informally be stated as follows:

1. To decrypt a message encrypted using PK during period i, only key $sk_{\langle i \rangle}$ is needed.

2. Given key sk_w, it is possible to efficiently derive keys sk_{w0} and sk_{w1}.

3. Given PK and i, and *without* sk_w for all prefixes w of $\langle i \rangle$, it is infeasible to derive $sk_{\langle i \rangle}$ and furthermore infeasible to decrypt messages encrypted during period i.

The formal statements corresponding to these requirements will become clear from the detailed description below.

Once we have a scheme satisfying the above requirements, we immediately obtain an efficient construction of a forward-secure encryption scheme.[1] For a given period i, let $i_0 i_1 \cdots i_t = \langle i \rangle$, where $i_0 = \varepsilon$ and $i_1, \ldots, i_t \in \{0,1\}$. The secret key SK_i for period i will consist of (1) $sk_{\langle i \rangle}$ and also (2) $\{sk_{i_0 \cdots i_{k-1} 1}\}$ for all $1 \le k \le t$ such that $i_k = 0$. Clearly, this allows for correct decryption of ciphertexts transmitted during the appropriate period. Furthermore, key updates can be done as follows: At the end of period i, key $sk_{\langle i \rangle}$ is erased and — as can be easily verified — the remainder of the keys can be updated as required (more detail is provided below). The above-stated requirements essentially imply the forward-security of this scheme.

One may notice that the requirements listed above immediately give rise to a hierarchical identity-based encryption scheme (HIBE) as well [15, 13]. Indeed, one can interpret our results as showing a generic transformation from any HIBE to a forward-secure PKE; we explore this connection further in the full version of this paper.

3.2 The Details

As mentioned in the previous section, an HIBE may be used toward the construction of a forward-secure PKE. The construction we present here is based on the HIBE suggested by [13] (the 2-HIBE of [15] is not suited for our purpose). We note that the proof of forward-security for our construction does not immediately follow from the results of [13]. In particular, for an adaptive adversary — as considered here — the HIBE of [13] is proven secure only for constant t (and hence constant N) whereas we give a proof of security for $t = \Theta(\log k)$ thereby allowing $N = \mathsf{poly}(k)$. Furthermore, because we do not require "full" security in the sense of HIBE we give a simpler proof and tighter security reduction.

Let $\mathcal{IG}$ be a BDH parameter generator for which the BDH assumption holds. We now present the details of the scheme.

$\mathsf{Gen}(1^k, N)$ does the following:

1. $\mathcal{IG}(1^k)$ is run to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of order q and bilinear map $\hat{e}$.
2. A random generator $P \leftarrow \mathbb{G}_1$ is selected along with random $s_\varepsilon \leftarrow \mathbb{Z}_q$. Set $Q = s_\varepsilon P$.
3. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$.
4. Set $S_0 = s_\varepsilon H_1(0)$ and $S_1 = s_\varepsilon H_1(1)$. Set $sk_0 = (S_0, \emptyset)$ and $sk_1 = (S_1, \emptyset)$. Using sk_0, recursively apply algorithm $\mathsf{Extract}$ (defined below) to generate keys sk_{01}, $sk_{001}, \ldots$, $sk_{0^{t-1}1}$, sk_{0^t}.
5. Store $SK_0 = (sk_{0^t}, \{sk_1, sk_{01}, \ldots, sk_{0^{t-1}1}\})$. Erase all other information.

We furthermore assume that hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ are defined, either by Gen or else as part of the specification of the scheme. These hash functions will be treated as random oracles in the analysis.

[1] This was noted in the context of digital signatures as well [4].

$\mathsf{Extract}(sk_w, w)$ takes as input the secret key associated with node w and outputs the secret keys for nodes $w0$ and $w1$. It runs as follows:

1. Parse w as $w_1 \cdots w_\ell$ where $|w| = \ell$. Parse sk_w as $(S_w, \mathcal{Q}_w)$ where $S_w \in \mathbb{G}_1$ and $\mathcal{Q}_w = (Q_{w_1}, \ldots, Q_{w_1 \cdots w_{\ell-1}}) \in \mathbb{G}_1^{\ell-1}$.
2. Choose random $s_w \in \mathbb{Z}_q$. Set $S_{w0} = S_w + s_w H_1(w0)$ and $S_{w1} = S_w + s_w H_1(w1)$. Set $Q_{w_1 \cdots w_\ell} = s_w P$ and $\mathcal{Q}_{w0} = \mathcal{Q}_{w1} = (Q_{w_1}, \ldots, Q_{w_1 \cdots w_\ell})$.
3. Output $sk_{w0} = (S_{w0}, \mathcal{Q}_{w0})$ and $sk_{w1} = (S_{w1}, \mathcal{Q}_{w1})$.

$\mathsf{Upd}(SK_i, i+1)$ (where $i < N-1$) does the following:

1. Parse $\langle i \rangle$ as $i_0 i_1 \cdots i_t$ where $i_0 = \varepsilon$. Parse SK_i as $(sk_{\langle i \rangle}, \{sk_{i_0 \cdots i_{k-1} 1}\}_{i_k = 0})$. Erase $sk_{\langle i \rangle}$.
2. We distinguish two cases. If $i_t = 0$, simply output the remaining keys as the key SK_{i+1} for the next period. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$ (such $\tilde{k}$ must exist since $i < N-1$). Let $i' = i_0 \cdots i_{\tilde{k}-1} 1$. Using $sk_{i'}$ (which is included as part of SK_i), recursively apply algorithm $\mathsf{Extract}$ to generate keys $sk_{i'1}, sk_{i'01}, \ldots, sk_{i'0^{t-\tilde{k}-1}1}, sk_{i'0^{t-\tilde{k}}}$. Erase $sk_{i'}$ and output the remaining keys as SK_{i+1}.

It can be easily verified that the key SK_{i+1} that is output has the correct form.

$\mathsf{Enc}_{PK}(i, M)$ (where $M \in \{0,1\}^n$) does the following:

1. Let $i_1 \cdots i_t = \langle i \rangle$. Select random $r \leftarrow \mathbb{Z}_q$.
2. Output $\langle i, C \rangle$ where $C = (rP, rH_1(i_1 i_2), \ldots, rH_1(i_1 \cdots i_t), M \oplus H_2(\hat{e}(Q, H_1(i_1))^r))$

$\mathsf{Dec}_{SK_i}(\langle i, C \rangle)$ does the following:

1. Parse $\langle i \rangle$ as $i_1 \cdots i_t$. Parse SK_i as $(sk_{\langle i \rangle}, \{sk_{i_0 \cdots i_{k-1} 1}\}_{i_k = 0})$ and $sk_{\langle i \rangle}$ as $(S_{\langle i \rangle}, \mathcal{Q}_{\langle i \rangle})$ where $\mathcal{Q}_{\langle i \rangle} = (Q_{i_1}, \ldots, Q_{i_1 \cdots i_{t-1}})$. Parse C as $(U_0, U_2, \ldots, U_t, V)$.
2. Compute
$$M = V \oplus H_2\left(\frac{\hat{e}(U_0, S_{\langle i \rangle})}{\Pi_{k=2}^t \hat{e}(Q_{i_1 \cdots i_{k-1}}, U_k)} \right).$$

We now verify that decryption is performed correctly. When encrypting, we have $\hat{e}(Q, H_1(i_1))^r = \hat{e}(P, H_1(i_1))^{rs_\varepsilon}$. When decrypting, we have $U_0 = rP$, $U_2 = rH_1(i_1 i_2), \ldots, U_t = rH_1(i_1 \cdots i_t)$ so that

$$\begin{aligned}
\frac{\hat{e}(U_0, S_{\langle i \rangle})}{\Pi_{k=2}^t \hat{e}(Q_{i_1 \cdots i_{k-1}}, U_k)} &= \frac{\hat{e}\left(rP, s_\varepsilon H_1(i_1) + \sum_{k=2}^t s_{i_1 \cdots i_{k-1}} H_1(i_1 \cdots i_k)\right)}{\prod_{k=2}^t \hat{e}\left(s_{i_1 \cdots i_{k-1}} P, rH_1(i_1 \cdots i_k)\right)} \\
&= \frac{\hat{e}(P, H_1(i_1))^{rs_\varepsilon} \cdot \prod_{k=2}^t \hat{e}\left(P, H_1(i_1 \cdots i_k)\right)^{rs_{i_1 \cdots i_{k-1}}}}{\prod_{k=2}^t \hat{e}\left(P, H_1(i_1 \cdots i_k)\right)^{rs_{i_1 \cdots i_{k-1}}}} \\
&= \hat{e}(P, H_1(i_1))^{rs_\varepsilon}
\end{aligned}$$

and thus decryption succeeds.

The security of the above scheme is stated in the following theorem:

Theorem 1 *Suppose there is an adversary A that has advantage δ against the above scheme in the sense of* $\mathsf{fs\text{-}OWE}$ *(cf. Remark 2) and that makes at most q_{H_2} hash queries to H_2. Then there is an algorithm A' that solves the BDH problem with respect to $\mathcal{IG}$ with probability at least $(\delta - 2^{-n})/N \cdot q_{H_2}$*

A proof of Theorem 1 appears in Appendix A.

In particular, the theorem implies that as long as $\mathcal{IG}$ satisfies the BDH assumption, then the above scheme is $\mathsf{fs\text{-}OWE}$ whenever $N = \mathsf{poly}(k)$. We note that the loss of a factor of N in the security reduction (implying the "limitation" $N = \mathsf{poly}(k)$) is also present in most previous work on forward-secure primitives (e.g., [4, 2, 16]). However, we can in fact improve upon this and construct a modified scheme whose security depends on *the number of periods elapsed thus far.* We can further modify this scheme so as to support an "unbounded" number of time periods (i.e., the number of time periods does not need to be known at the time of key generation).[2] We defer details to the final version.

As in [6, 15, 13], we may apply the transformation due to Fujisaki and Okamoto [12] to obtain a scheme which is forward-secure under adaptive chosen-ciphertext attacks. Details and a full proof (which is non-trivial and does not follow immediately from the results of [12]) will appear in the final version.

3.3 Analysis of Parameters

We briefly justify the claims given in Table 1. Key generation requires t invocations of $\mathsf{Extract}$ in addition to $\mathcal{O}(1)$ other operations. Each invocation of $\mathsf{Extract}$ runs in $\mathcal{O}(1)$ time, showing that the entire key generation process requires time $\mathcal{O}(\log N)$. Encryption time, decryption time, and ciphertext length are all clearly $\mathcal{O}(\log N)$. The key update time is $\mathcal{O}(\log N)$ *in the worst case*; amortizing over all time periods results in complexity $\mathcal{O}(1)$. Finally, note that during any time period i the secret key SK_i consists of at most $\log N$ "node" secret keys sk_w; furthermore, each node secret key is of size $\mathcal{O}(\log N)$. This implies that the total secret storage is $\mathcal{O}(\log^2 N)$.

Acknowledgments

We are very grateful to Craig Gentry for helpful discussions regarding [13] as well as a preliminary version of subsequent work.

References

[1] M. Abdalla, S. Miner, and C. Namprempre. Forward-Secure Threshold Signature Schemes. RSA '01.

[2] M. Abdalla and L. Reyzin. A New Forward-Secure Digital Signature Scheme. Asiacrypt '00.

[3] R. Anderson. Two Remarks on Public-Key Cryptology. Invited lecture, CCCS '97. Available at `http://www.cl.cam.ac.uk/users/rja14/`.

[2] Similar results were previously shown by [19] in the context of forward-secure digital signatures.

[4] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. Crypto '99.

[5] M. Bellare and B. Yee. Forward Security in Private-Key Cryptography. Manuscript, Nov. 2001. Available at `http://eprint.iacr.org`.

[6] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. Crypto '01. Full version to appear in *SIAM J. Computing* and available at `http://eprint.iacr.org/2001/090/`.

[7] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. Asiacrypt '01.

[8] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. Draft. Presented at Crypto 2002 rump session.

[9] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89.

[10] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public-Key Cryptosystems. Eurocrypt '02.

[11] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. Manuscript, April 2002.

[12] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. Crypto '99.

[13] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. Manuscript, May 2002. Available at `http://eprint.iacr.org`.

[14] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public-Key and Signature Schemes. CCCS '97.

[15] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. Eurocrypt '02.

[16] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. Crypto '01.

[17] A. Joux. A One-Round Protocol for Tri-Partite Diffie Hellman. ANTS '00.

[18] H. Krawczyk. Simple Forward-Secure Signature From any Signature Scheme. CCCS '00.

[19] T. Malkin, D. Micciancio, and S. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods. Eurocrypt '02.

[20] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91.

[21] A. Shamir. How to Share a Secret. *Comm. ACM*, 22(11):612–613, 1979.

[22] W. Tzeng and Z. Tzeng. Robust Forward-Secure Digital Signatures with Proactive Security. PKC '01.

[23] E. Verheul. Self-Blindable Credential Certificates from the Weil Pairing. Asiacrypt '01.

A Proof of Theorem 1

Our proof is largely similar to that of [13] except that, because we work in a different context, we are able to simplify some details. As mentioned earlier, our results are not immediately implied by those of [13] since (in their setting) they are unable to handle an adaptive adversary when $t = \Omega(1)$. In our context, we are able to achieve security against an adaptive adversary even when $t = \mathcal{O}(\log k)$.

Assume an adversary A who has advantage δ in attacking the scheme of Section 3 in the sense of fs-OWE. We show how to construct an adversary A' solving the BDH problem with probability at least $(\delta - 2^{-n})/q_{H_2}N$.

Adversary A' is given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ as output by $\mathcal{IG}(1^k)$, and is additionally given random elements $P, Q = s_\varepsilon P, P' = bP$, and $U_0 = cP$. The goal of A' is to output $\hat{e}(P,P)^{s_\varepsilon bc}$. A' will simulate an instance of the ke-PKE for adversary A. First, A' sets $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$ and gives PK to A. Next, A' guesses a random index $i^* \in \{0, \ldots, N-1\}$ (this represents a guess of the period for which A will request a challenge). Let $\langle i \rangle = i_1^* \cdots i_t^*$ and $i_0^* = \varepsilon$.

To answer the hash queries of A, algorithm A' maintains lists H_1^{list} and H_2^{list}. To begin, H_2^{list} will be empty. H_1^{list} is prepared by first having A' select random $x_2, \ldots, x_t \in \mathbb{Z}_q$ and then storing the tuples $(i_1^*,\ P')$, $(i_1^* i_2^*,\ x_2 P), \ldots, (i_1^* \cdots i_t^*,\ x_t P)$ in H_1^{list}. Next, A' generates "node keys" $\{sk_{i_0^* \cdots i_{k-1}^* 1}\}_{i_k^*=0}$ as follows:

1. If $i_1^* = 0$, choose random $y_1 \in \mathbb{Z}_q$. Store $(1, y_1 P)$ in H_1^{list} and set $sk_1 = (y_1 Q, \emptyset)$.
2. Select random $s'_{i_1^*}, s'_{i_1^* i_2^*}, \ldots, s'_{i_1^* \cdots i_{t-1}^*} \in \mathbb{Z}_q$.
3. For each $2 \le k \le t$ such that $i_k^* = 0$:
   (a) Choose random $y_k \in \mathbb{Z}_q$ and store $\left(i_1^* \cdots i_{k-1}^* 1,\ y_k P - (s'_{i_1^* \cdots i_{k-1}^*})^{-1} P'\right)$ in H_1^{list}.
   (b) Set $sk_{i_1^* \cdots i_{k-1}^* 1} = \left(s'_{i_1^* \cdots i_{k-1}^*} y_k Q + \sum_{\ell=2}^{k-1} s'_{i_1^* \cdots i_{\ell-1}^*} x_\ell Q,\ \left(s'_{i_1^*} Q,\ \ldots,\ s'_{i_1^* \cdots i_{k-1}^*} Q\right)\right)$.[3]

A' will respond to hash queries of A in the obvious way. If A queries $H_b(X)$, then A' checks whether there is a tuple of the form (X, Y) in H_b^{list}. If so, the value Y is returned. Otherwise, A' chooses random Y from the appropriate range, stores (X, Y) in H_b^{list}, and returns Y.

We now verify that the node keys created above have the correct distribution. In case $i_1^* = 0$, note that $y_1 Q = s_\varepsilon y_1 P = s_\varepsilon H_1(1)$ so that sk_1 is of the correct form. For $2 \le k \le t$, define $s_{i_1^* \cdots i_{k-1}^*} = s_\varepsilon s'_{i_1^* \cdots i_{k-1}^*}$. We then have:

$$\begin{aligned} s'_{i_1^* \cdots i_{k-1}^*} y_k Q + \sum_{\ell=2}^{k-1} s'_{i_1^* \cdots i_{\ell-1}^*} x_\ell Q &= s_{i_1^* \cdots i_{k-1}^*} y_k P - s_\varepsilon P' + s_\varepsilon P' + \sum_{\ell=2}^{k-1} s_{i_1^* \cdots i_{\ell-1}^*} x_\ell P \\ &= s_{i_1^* \cdots i_{k-1}^*} (y_k P - (s'_{i_1^* \cdots i_{k-1}^*})^{-1} P') + s_\varepsilon P' \\ &\quad + \sum_{\ell=2}^{k-1} s_{i_1^* \cdots i_{\ell-1}^*} x_\ell P \end{aligned}$$

[3] For $k = 2$ the upper limit of the summation is less than the lower limit; by convention, we let the sum in this case be 0.

$$
\begin{aligned}
= \quad & s_\varepsilon H_1(i_1^*) + \left(\sum_{\ell=1}^{k-2} s_{i_1^* \cdots i_\ell^*} H_1(i_1^* \cdots i_{\ell+1}^*) \right) \\
& + s_{i_1^* \cdots i_{k-1}^*} H_1(i_1^* \cdots i_{k-1}^* 1).
\end{aligned}
$$

Furthermore, $s'_{i_1^* \cdots i_{k-1}^*} Q = s_{i_1^* \cdots i_{k-1}^*} P$. Thus, $sk_{i_1^* \cdots i_{k-1}^* 1}$ has the correct form for all k such that $i_k^* = 0$. Using these node keys, A' can derive key SK_{i^*+1} with the correct distribution; this implies that A' can correctly respond to query $\mathsf{breakin}(i)$ of A for any $i > i^*$.

A' now runs A, responding to hash queries as described previously. If A makes a query $\mathsf{breakin}(i)$ for $i \leq i^*$ or a query $\mathsf{challenge}(j)$ for $j \neq i^*$, then A' aborts. Otherwise, A' responds to the $\mathsf{breakin}(i)$ query by generating the appropriate node keys and giving these to A (as mentioned previously, A' can do this whenever $i > i^*$). A' responds to the query $\mathsf{challenge}(i^*)$ with $\langle i^*, C \rangle$, where $C = (U_0, x_2 U_0, \ldots, x_t U_0, V)$ and V is selected at random from $\{0,1\}^n$. Note that, as long as A' does not abort, this results in a perfect simulation of the view of A.

Eventually (assuming A' does not abort), A outputs a guess M in an attempt to break the one-way security of the scheme. At this point, A' picks a random tuple (X, Y) from H_2^{list} and outputs X. Let $X^* = \hat{e}(P,P)^{s_\varepsilon bc} = \hat{e}(Q, P')^c$. Let query be the event that, at the end of the simulation, X^* appears as the first element of some tuple in H_2^{list} and let $\mathsf{correct}$ be the probability that A succeeds in correctly guessing M. By assumption, $\Pr[\mathsf{correct}] = \delta$. Since H_2 is modeled as a random oracle, we have $\Pr[\mathsf{correct} | \overline{\mathsf{query}}] = 2^{-n}$. Therefore:

$$
\begin{aligned}
\delta &= \Pr[\mathsf{correct}|\mathsf{query}] \Pr[\mathsf{query}] + \Pr[\mathsf{correct}|\overline{\mathsf{query}}] \Pr[\overline{\mathsf{query}}] \\
&\leq \Pr[\mathsf{query}] + 2^{-n}(1 - \Pr[\mathsf{query}])
\end{aligned}
$$

so that $\Pr[\mathsf{query}] \geq \delta - 2^{-n}$. The probability that A' does not abort is exactly $1/N$. Assuming A' does not abort, A' outputs the correct answer with probability $\Pr[\mathsf{query}]/q_{H_2}$. Theorem 1 follows.