This Extended Abstract appears in
Proceedings of the 9th International Conference on Security and Cryptography for Networks (SCN '14)
3 – 5 September 2014, Amalfi, Italy – Michel Abdalla and Roberto de Prisco Eds., Springer-Verlag, LNCS 8642, pages 21–39.

Forward Secure Non-Interactive Key Exchange

David Pointcheval[1] and Olivier Sanders[1,2]

[1] École normale supérieure, CNRS & INRIA, Paris, France
[2] Orange Labs, Applied Crypto Group, Caen, France

Abstract. Exposure of secret keys is a major concern when cryptographic protocols are implemented on weakly secure devices. Forward security is thus a way to mitigate damages when such an event occurs. In a forward-secure scheme, the public key is indeed fixed while the secret key is updated with a one-way process at regular time periods so that security of the scheme is ensured for any period prior to the exposure, since previous secret keys cannot be recovered from the corrupted one. Efficient constructions have been proposed for digital signatures or public-key encryption schemes, but none for non-interactive key exchange protocols, while the non-interactivity makes them quite vulnerable since the public information cannot evolve from an execution to another one.
In this paper we present a forward-secure non-interactive key exchange scheme with sub-linear complexity in the number of time periods. Our protocol is described using generic *leveled* multilinear maps, but we show that it is compatible with the recently introduced candidates for such maps. We also discuss various security models for this primitive and prove that our scheme fulfills them, under standard assumptions.

Keywords: forward security, non-interactive key exchange, multilinear map.

1 Introduction

1.1 Non-Interactive Key Exchange

The famous interactive key exchange protocol introduced in 1976 in the seminal paper [DH76] by Diffie and Hellman can be turned into a simple and quite efficient non-interactive key exchange (NIKE) scheme: it enables two parties, who have first agreed on some parameters, to share a common secret without exchanging any additional messages but just their public keys. More precisely, the parameters simply consist of a group $\mathbb{G}$ of prime order p along with a generator $g \in \mathbb{G}$. When Alice, whose secret/public keys pair is $(x, X = g^x)$ for some $x \in \mathbb{Z}_p$, wants to share a secret with Bob, whose public key is $Y = g^y$, she computes $K = Y^x$, which value can be recovered by Bob by computing X^y. However, eavesdroppers have no clue about this value, because of the intractability of the Diffie-Hellman problem. Hashing the resulting secret K along with both identities even leads to a provably secure scheme, according to the expected properties for a NIKE scheme, and this scheme is remarkably efficient. Indeed, both secret and public keys consist of one element and sharing a secret only requires one exponentiation from each user.

A first basic security model for NIKE has been provided by Bernstein [Ber06]. Thereafter, Cash, Kiltz and Shoup [CKS08] enhanced it, allowing dishonestly generated public keys. This models the real-life situation where public keys are published by users, without certification, or with a weak certification only (when the certification authority does not check the knowledge of the associated secret key, but just the identity of the owner of the public key). However, Freire *et al* [FHKP13] pointed out some weaknesses in their model such as the inability of the adversary to corrupt honest users, and thus get honestly generated secret keys or shared keys between two honest entities. They proposed the *dishonest-key registration model*, as the strongest security model, together with a scheme in a pairing-friendly setting, secure in the standard model.

Sakai, Oghishi and Kasahara [SOK00] proposed the first Identity-based NIKE (Id-NIKE) scheme, later formalized and proven secure by Dupont and Enge in an ad-hoc security model [DE06]. The above concerns about the Certification Authority, and dishonestly generated public keys, are irrelevant in the identity-based setting, however, again, the lack of oracle access to previous shared keys was noticed as a potential weakness by Paterson and Srinivasan [PS09]. They thus fixed the previous model and explored the relationships between Id-NIKE and Identity-Based Encryption (IBE). Moreover, they proposed constructions, using trapdoor discrete log groups, whose instantiations suffer from the high

computational cost of the `Extract` algorithm (to get secret keys from identities), with security in the random oracle model. Recently, Freire *et al* [FHPS13] provided the first Id-NIKE and Hierarchical Id-NIKE schemes secure against corruptions in the standard model.

1.2 Forward Security

As for most of cryptographic protocols, the main threat against a NIKE scheme is exposure of users' secret keys since, contrarily to interactive key exchange protocols which can still provide some security in this case, all the session keys between the corrupted user and any other user get immediately leaked. Leakage of a secret key is therefore a major issue for all users, not only for the corrupted one. A classical solution to prevent leakage is to distribute the secret across multiple servers via secret sharing. However, this is not compatible with the goal of *non-interactive* key exchange which is to limit communications between the different parties. Anderson [And97] thus suggested *forward security* to mitigate damages caused by key exposure: the lifetime of a system is now divided into T time periods, the secret keys evolving with time. More precisely, at any time period i, each user owns a secret key sk_i which he can use as usual, but also to derive his secret key sk_{i+1}, for the next time period. However, forward security requires that an adversary being able to recover sk_i is unable to compromise the security of any previous time period: the evolving process from sk_i to sk_{i+1} has to be one-way. In his talk, Anderson proposed a non trivial solution, but constructing protocols whose parameters were sub-linear in the number of periods remained a challenge. The case of digital signatures was first addressed by Bellare and Miner [BM99] which provided a security model but also different constructions, one of them achieving constant key-size (*i.e.* independent of T). Then, many other papers followed [AR00,IR01,KR02,BSSW06,ABP13], most of them providing schemes in the RSA setting. The case of public-key encryption has later been addressed by Canetti, Halevi and Katz [CHK07] whose construction in a pairing-friendly setting has complexity logarithmic in T only.

Although the case of forward-secure NIKE was mentioned in [And97], the problem of constructing a scheme with sub-linear complexity has still remained open. One could think that the ideas used to construct forward-secure signature or encryption schemes can lead to forward-secure NIKE schemes, however, this does not seem to be the case for the reasons we describe below. We here make a distinction between constructions in the RSA setting [BM99,AR00,IR01,ABP13] and the ones in a pairing-friendly setting [BSSW06,CHK07].

The first forward-secure schemes were proposed in the RSA setting: the key evolving process relies on the fact that exponentiation is a one-way function, even with a public exponent. Informally, the underlying idea is to set the public key as $Z = S^{e_1 \cdot \ldots \cdot e_T}$ for T public exponents $e_1, ..., e_T$ (we may have $e_i = e_j$) and a secret element S. At each time period i, the user will prove knowledge of an $(e_i \cdot ... \cdot e_T)$-th root of Z (thus sk_i is $S^{e_1 \cdot \ldots \cdot e_{i-1}}$), such a proof leading to an efficient signature scheme by using the Fiat-Shamir heuristic [FS86]. Updating the secret key simply consists in computing $sk_i^{e_i}$ and so does not require any randomness which would be convenient for constructing a NIKE scheme. However, while assuming that Alice knows some n-th root of a public element Z_A and that Bob knows an m-th root of a public element Z_B, computing a common secret between Alice and Bob is far from being obvious. Therefore, the RSA setting unfortunately seems to be more suitable for signatures than for NIKE schemes.

Since the seminal paper from Joux [Jou00], pairings have been widely used in cryptography, their properties allowing to solve open problems such as to construct an efficient identity-based encryption scheme [BF01]. In [CHK07], Canetti, Katz and Halevi used them to propose a forward-secure encryption scheme with logarithmic complexity in the number of time periods. However, since the involved groups are of prime order, exponentiation with public exponent is no longer a one-way function. The update algorithm is then more complex and involves randomness which makes sharing a common secret more difficult since non-interactivity of the primitive implies that Bob cannot get information about the random values used by Alice. The signature scheme of Boyen *et al* [BSSW06] is quite similar, therefore the underlying idea of constructions in a pairing-friendly setting does not seem to suit the NIKE case either. However, we emphasize that the randomness used to update the secret key is not

necessarily incompatible with NIKE but the protocol must ensure that the common secret shared by Alice and Bob is independent of it.

1.3 Achievements

The lack of forward-secure NIKE scheme with sub-linear complexity could be explained by the limitations of cryptographic tools known until recently. As with pairings a decade ago, the recent candidates for multilinear maps proposed by Garg, Gentry and Halevi [GGH13] and Coron, Lepoint and Tibouchi [CLT13] offer new functionalities allowing to achieve constructions previously impossible. An example is provided in [FHPS13] where the authors used them to propose the first Id-NIKE scheme secure in the standard model.

In this work we prove that constructing a forward-secure NIKE scheme is also possible by using multilinear maps. Our scheme shares some similarities with tree-based forward-secure schemes since we also associate time periods with all nodes of the tree. But the construction manages to handle both evolution of secret keys and key exchange with the tree. It also provides some flexibility with the number of levels of the multilinear map, since whatever the number of time periods a bilinear map can be enough, at the cost of a larger secret key, while a smaller secret key will require a multilinear map with a higher number of levels. In addition, our construction is compatible with multilinear maps from [GGH13] and [CLT13], but requires some modifications that we describe in this paper. We also formally define two security models for forward-secure NIKE and prove that our scheme achieves the strongest one under a conventional assumption in the standard model.

1.4 Organization

In the next section, we recall the definition of generic leveled-multilinear maps and some of the differences with their approximations proposed in [GGH13] and [CLT13]. Section 3 describes a security model for forward-secure NIKE. We present a protocol using binary tree in Section 4 and then discuss the necessary adjustments to suit existing implementations of multilinear maps. We then show how to generalize the underlying idea of the previous protocol to get a trade-off between the size of the secret key and the number of levels of the multilinear map.

2 Leveled Multilinear Maps

Boneh and Silverberg [BS03] defined n-linear maps as non-degenerate maps e from $\mathbb{G}_1^n$ to $\mathbb{G}_2$ (where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of same order) such that, for all $g_1, ..., g_n \in \mathbb{G}_1$ and $a_1, ..., a_n \in \mathbb{Z}$, $e(g_1^{a_1}, ..., g_n^{a_n}) = e(g_1, ..., g_n)^{a_1 \cdots a_n}$. The candidate multilinear map proposed by Garg, Gentry and Halevi [GGH13] actually yields a richer structure since it is now possible to multiply any (bounded) subset of encodings instead of n at a time. As in [HSW13], such maps will be denoted *leveled* multilinear maps. We recall the formal definition of generic n-leveled multilinear groups.

Leveled Multilinear Maps. Generic n-leveled multilinear groups consist of n cyclic groups $\mathbb{G}_1, ..., \mathbb{G}_n$ of prime order p, along with bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$ for $i, j \geq 1$ and $i+j \leq n$ such that, for all $g_i \in \mathbb{G}_i$, $g_j \in \mathbb{G}_j$ and $a, b \in \mathbb{Z}_p$, $e_{i,j}(g_i^a, g_j^b) = e_{i,j}(g_i, g_j)^{a \cdot b}$. In the following we will write e instead of $e_{i,j}$ when i and j are obvious and $e(g_1, g_2, \ldots, g_n)$ instead of $e(g_1, e(g_2, \ldots g_n) \ldots))$.

The *graded encoding schemes* proposed in [GGH13] and [CLT13] are only approximations of such leveled multilinear maps. One of the main differences is that group elements have many possible representations called *encodings*. An encoding can be re-randomized but at the cost of introducing some noise. Such a randomization, performed using the Rerand algorithm, is sometimes necessary to prevent recovering of secret values. Indeed, let c be a secret level-zero encoding and g a public level-1 encoding. The level-1 encoding $y = c \cdot g$ (which is the equivalent of g^c in conventional groups) cannot

be directly published since anyone will be able to recover c by computing $y.g^{-1}$, it must first be re-randomized into a new level-1 encoding $y' \leftarrow \mathsf{Rerand}(y)$. All these randomizations could be an obstacle for sharing a common secret, however, it is possible to extract, using the $\mathsf{Extract}$ algorithm, a *canonical* bit string which depends on the group element and not on its encoding, meaning that two encodings of the same element will give the same extracted string. Then the security of our protocols relies on the following assumption.

The n-Multilinear Decisional Diffie-Hellman (n-MDDH) Assumption.
Given $(g, g^{x_1}, ..., g^{x_{n+1}}, G) \in \mathbb{G}_1^{n+2} \times \mathbb{G}_n$, it is hard to decide whether $G = e(g^{x_1}, ..., g^{x_n})^{x_{n+1}}$ or not.

3 Forward-Secure Non-interactive Key Exchange and Security Model

3.1 Syntax

Following [CKS08] and [FHKP13], a forward-secure non-interactive key-exchange is defined by the following algorithms along with an identity space $\mathcal{IDS}$ and a shared key space $\mathcal{SHK}$. Identities are used to track which public keys are associated with which users but we are not in the identity-based setting.

- $\mathtt{Setup}(1^k, T)$: On inputs a security parameter k and a number of time periods T, this probabilistic algorithm outputs *params*, a set of system parameters that are implicit to the other algorithms. The current time period t^* is initially set to 1;
- $\mathtt{Keygen}(ID)$: On input an identity $ID \in \mathcal{IDS}$ this probabilistic algorithm outputs a public key/secret key pair (pk, sk_{t^*}), for the current time period t^*. We assume that the secret keys implicitly contain the time periods, hence the subscripts;
- $\mathtt{Update}(sk_t)$: This algorithm takes as inputs the secret key sk_t at some period t (implicitly included in sk_t) and outputs the new secret key sk_{t+1} for the next time period, if $t < T$. If $t = T$ then the secret key is erased and there is no new key;
- $\mathtt{Sharekey}(ID_A, pk^{(A)}, ID_B, sk_t^{(B)})$: On inputs an identity ID_A, associated with a public key $pk^{(A)}$ and a secret key $sk_t^{(B)}$ with identity ID_B, outputs either a shared key $shk_t^{AB} \in \mathcal{SHK}$ or a failure symbol $\perp$. This algorithm outputs $\perp$ if $ID_A = ID_B$. Since the secret key sk_t contains the time period, the shared key shk_t^{AB} is also specific to that time period t.

Correctness requires that, for any pair (ID_A, ID_B), if the secret keys $sk_t^{(A)}$ and $sk_t^{(B)}$ indeed correspond to the same time period t:

$$\mathtt{Sharekey}(ID_A, pk^{(A)}, ID_B, sk_t^{(B)}) = \mathtt{Sharekey}(ID_B, pk^{(B)}, ID_A, sk_t^{(A)}).$$

As in most of the forward-secure primitives, we provide the number of time periods to the $\mathtt{Setup}$ algorithm, because the parameters depend on it. In practice, one can take T large enough. Note that $T = 2^{15}$ is enough to enumerate one-day time-periods for one century.

3.2 Security Model

We define the security of a forward-secure non-interactive key exchange through a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Our security model makes use of the following oracles:

- $\mathcal{O}\mathtt{RegHon}(ID)$ is an oracle used by $\mathcal{A}$ to register a new honest user ID at the initial time period. The challenger runs the $\mathtt{Keygen}$ algorithm with 1 as the current time period, returns the public key pk to $\mathcal{A}$ and records $(ID, sk_1, pk, \mathsf{honest})$. This implicitly defines all the secret keys $sk_2, \ldots, sk_T$;
- $\mathcal{O}\mathtt{RegCor}(ID, pk)$ is an oracle used by $\mathcal{A}$ to register a new corrupted user ID with public key pk. The challenger then records the tuple $(ID, -, pk, \mathsf{corrupted})$.

- $\mathcal{O}\texttt{Breakin}(ID, t)$ is an oracle used by $\mathcal{A}$ to get the ID's secret key at the time period t. The challenger looks for a tuple $(ID, sk_1, pk, \mathsf{honest})$. If there is a match, then it returns sk_t. Else, it returns $\perp$.
- $\mathcal{O}\texttt{Reveal}(ID_A, ID_B, t)$ is an oracle used by $\mathcal{A}$ to get the shared key $shk_t^{(AB)}$ between ID_A and ID_B for the time period t. If both ID_A and ID_B are corrupted then $\mathcal{C}$ returns $\perp$. Else, it runs the $\texttt{Sharekey}$ algorithm with the secret key of one of the honest users for the appropriate time period and the public key of the other user and returns $shk_t^{(AB)}$.

A non-interactive key exchange is forward-secure if, for any adversary $\mathcal{A}$ and any security parameter k, the advantage $\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fs}}(k) = 1] - \frac{1}{2}$ is negligible in k, where $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{fs}}(k)$ is defined as follows:

1. $params \leftarrow \texttt{Setup}(1^k, T)$
2. $(ID_A, ID_B, t^*) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{RegHon},\mathcal{O}\texttt{RegCor},\mathcal{O}\texttt{Breakin},\mathcal{O}\texttt{Reveal}}(params)$
3. $b \xleftarrow{\$} \{0, 1\}$
4. If $b = 0$ then $shk_{t^*}^{(AB)} \leftarrow \texttt{Sharekey}(ID_A, pk^{(A)}, ID_B, sk_{t^*}^{(B)})$
5. Else, $shk_{t^*}^{(AB)} \xleftarrow{\$} \mathcal{SHK}$
6. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\texttt{RegHon},\mathcal{O}\texttt{RegCor},\mathcal{O}\texttt{Breakin},\mathcal{O}\texttt{Reveal}}(shk_{t^*}^{(AB)})$
7. If ID_A or ID_B is corrupted then return 0
8. If an $\mathcal{O}\texttt{Breakin}$-query has been asked on $ID \in \{ID_A, ID_B\}$ with $t \leq t^*$ then return 0
9. If an $\mathcal{O}\texttt{Reveal}$-query has been asked on (ID_A, ID_B, t^*) then return 0
10. If $b^* = b$ then return 1
11. Else, return 0

The adversary succeeds if it is able to distinguish a valid shared key between two users from a random element of the shared key space $\mathcal{SHK}$. To avoid trivial cases, the adversary is not allowed to corrupt the targeted users at a time period prior to t^* or to get the shared key between them at this time period. We emphasize that the adversary may corrupt any user (including ID_A or ID_B) for time periods $t > t^*$, which models the forward security.

Registration. The use of a Certification Authority (CA) is inevitable for protocols which are not in the ID-based settings, in order to link ID's and pk's. However, the assumptions made about the procedures followed by this entity differ according to each model.

- In the *registered-key model*, when a user wants to get his pubic key certified, the CA verifies, using a proof of knowledge, that the user actually knows the associated secret key. This enables the challenger (the simulator in the security proof) to extract the secret key and thus to answer every $\mathcal{O}\texttt{Reveal}$-queries involving corrupted users.
- Cash, Kiltz and Shoup [CKS08] considered a stronger model where the CA no longer requires such a proof of knowledge of the secret key. In [FHKP13], the authors named it the *dishonest-key registration model*, since the public keys are not checked anymore.

In the latter case, some $\mathcal{O}\texttt{Reveal}$ queries are not easy to answer, since none of the secret keys are known to the challenger/simulator. Hence the use of the Twin Diffie-Hellman [CKS08] which allows the challenger to check some consistency, in the random oracle model, under the sole CDH assumption. The requirement of the random oracle model has been more recently removed [FHKP13], by using chameleon hash functions [KR00]. To this end, they actually add some elements to the public key which provide a way for the challenger to recover the Diffie-Hellman value without knowledge of the corresponding secret keys. However, consistency still has to be checked, which is possible in the pairing settings only.

We can use the same approach, but with generic leveled multilinear maps, which are not provided with the existing implementations [GGH13,CLT13] of such maps. We explain the reasons in the Section 4.4.

3.3 Forward Security with Linear Complexity

A trivial solution, secure in our model, is to generate T independent keys pair (pk'_i, sk'_i) for any NIKE scheme and to set pk as $(pk'_1, \ldots, pk'_T)$ and sk_1 as $(sk'_1, \ldots, sk'_T)$. Updating the secret key $sk_i = (sk'_i, \ldots, sk'_T)$ simply consists in erasing the value sk'_i. To avoid the linear complexity of the public key we may use the following idea, similar to the one from [And97]: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$ be a bilinear setting, with groups of prime order p, and $g_1, \ldots, g_T \xleftarrow{\$} \mathbb{G}_1$, $g \xleftarrow{\$} \mathbb{G}_2$. To generate his public/secret key pair, a user randomly selects $x \xleftarrow{\$} \mathbb{Z}_p$ and sets $pk \leftarrow g^x$ and $sk_1 \leftarrow (g_1^x, \ldots, g_T^x)$. To update his secret key a user proceeds as in the previous solution, and to share a session key, at time period t, with another user whose public key is g^y, he computes $e(g_t^x, g^y)$. This scheme is correct and secure in the *registered-key model*. We can also avoid the linear complexity of the public parameters by setting $g_i \leftarrow H(t_i)$ for some cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. But the resulting scheme will now be secure in the *random oracle model* only, with both the public keys and public parameters of constant size, while the secret keys are linear in the number of time periods T. Our goal is now to achieve a sub-linear complexity for the secret keys too.

4 A Forward-Secure Non-Interactive Key Exchange Scheme

As in [HSW13], we first describe a version of our scheme, using generic leveled multilinear maps, that we prove secure in the *registered-key model*. We then explain how to achieve security in the *dishonest-key registration model* defined in [FHKP13] and discuss the necessary adjustments to suit existing multilinear maps [GGH13,CLT13] .

4.1 The protocol

Let $\mathcal{S}_n$ be the set of bitstrings of size smaller than n. We recall the lexicographic order on $\mathcal{S}_n$. Let $s = b_1 \cdots b_\ell$ and $s' = b'_1 \cdots b'_k$, with $\ell \leq k \leq n$, be two bitstrings, then:

- if $b_i = b'_i, \forall\ 1 \leq i \leq \ell$, then $s < s'$ if $\ell < k$, and $s = s'$ otherwise;
- else, let $j \leq \ell$ be such that $b_i = b'_i, \forall\ 1 \leq i < j$, but $b_j \neq b'_j$. If $b_j < b'_j$ then $s < s'$, else $s > s'$.

Each bitstring $s \in \mathcal{S}_n$ will now refer to a time period. Specifically, the i-th bitstring of $\mathcal{S}_n$ (considering that the empty string does not belong in the set) will refer to the i-th time period: the order is thus $0, 00, 000, \ldots, 0^n, 0^{n-1}1, 0^{n-2}1, 0^{n-2}10,\ 0^{n-2}11, \ldots, 1^n$, with $T = \#\mathcal{S}_n = 2^{n+1} - 2$ elements, and thus corresponding to $2^{n+1} - 2$ time periods.

Let $(\mathbb{G}_1, \ldots, \mathbb{G}_{n+1})$ be an $(n+1)$-leveled multilinear group setting of order p, the algorithms defining our forward-secure NIKE are described below:

- $\texttt{Setup}(1^k, n)$: This algorithm outputs the parameters $(g, g_1, ..., g_n) \xleftarrow{\$} \mathbb{G}_1^{n+1}$ along with $h_s \xleftarrow{\$} \mathbb{G}_1$ for each bitsring $s \in \mathcal{S}_n$. The public parameters *params* contain the $T + n + 1$ elements. In the following, for each $s = b_1 \cdots b_\ell \in \mathcal{S}_n$, we will denote $G_s = e(h_{b_1}, h_{b_1 b_2}, \ldots, h_{b_1 b_2 \cdots b_\ell}) \in \mathbb{G}_\ell$, where $e(h_b) = h_b \in \mathbb{G}_1$.
- $\texttt{Keygen}(ID)$: The user ID first selects $x \xleftarrow{\$} \mathbb{Z}_p$ and then outputs $pk \leftarrow g^x$ and the secret key of ID at the first time period, for $s = 0$, is $sk_0 \leftarrow \{h_1^x, h_0^x\}$. In the following, for each $s = b_1 \cdots b_\ell \in \mathcal{S}_n$, we denote $z_s^{(ID)} = G_s^x \in \mathbb{G}_\ell$.
- $\texttt{Update}(sk_s)$: Let ℓ be the length of $s = b_1 \cdots b_\ell$ and $\mathcal{I}_s$ be the set $\{1 \leq i \leq \ell : b_i = 0\}$. Then, sk_s can be parsed as $\cup_{i \in \mathcal{I}_s} \{z_{b_1 \cdots b_{i-1}1}\} \cup \{z_s\}$ (see the Correctness paragraph below) and the algorithm proceeds as follows:
  - if $\ell < n$, then the next bitstring is $s||0$, the algorithm computes $z_{s||0} \leftarrow e(z_s, h_{s||0})$, $z_{s||1} \leftarrow e(z_s, h_{s||1})$ and returns $sk_{s||0} \leftarrow (sk_s \setminus \{z_s\}) \cup \{z_{s||1}, z_{s||0}\}$;
  - if $\ell = n$, then we have $s = b_1 \cdots b_n$. If $s = 1^n$, then we have reached the last time period and the algorithm returns $\perp$. Else, let j be the greatest index such that $b_j = 0$, the next time period s^* is then $b_1 \cdots b_{j-1}1$. The algorithm then returns $sk_{s^*} \leftarrow \cup_{i \in \mathcal{I}_s, i \leq j} \{z_{b_1 \cdots b_{i-1}1}\} \subset sk_s$.
- $\texttt{Sharekey}(ID_A, pk^{(A)}, ID_B, sk_s^{(B)})$: This algorithm returns $\perp$ if $ID_A = ID_B$. Else, it outputs $shk_s^{AB} \leftarrow e(z_s^{(B)}, g_{\ell+1}, \ldots, g_n, pk^{(A)})$ where ℓ is the length of s.

Correctness. We first prove by induction that, for each time period s, $sk_s = \cup_{i\in\mathcal{I}_s}\{z_{b_1\cdots b_{i-1}1}\}\cup\{z_s\}$.

- For $s = 0$ we have $\mathcal{I}_s = \{0\}$ and $sk_0 = \{z_1\}\cup\{z_0\}$.
- If $\ell < n$ then the length of the next time period, $s^* = s||0$, is $\ell+1$ and $\mathcal{I}_{s^*} = \mathcal{I}_s\cup\{\ell+1\}$. The `Update` algorithm ensures that the next secret key sk_{s^*} is $\cup_{i\in\mathcal{I}_{s^*}}\{z_{b_1\cdots b_{i-1}1}\}\cup\{z_{s^*}\}$.
- If $\ell = n$, then the next time period is $s^* = b_1\cdots b_{j-1}1$, so $\mathcal{I}_{s^*} = \mathcal{I}_s\cap\{1,\ldots,j-1\}$. The new secret key $sk_{s^*} := \cup_{i\in\mathcal{I}_s, i\le j}\{z_{b_1\cdots b_{i-1}1}\} =$
  $\cup_{i\in\mathcal{I}_s, i<j}\{z_{b_1\cdots b_{i-1}1}\}\cup\{z_{s^*}\} = \cup_{i\in\mathcal{I}_{s^*}}\{z_{b_1\cdots b_{i-1}1}\}\cup\{z_{s^*}\}$ is then consistent.

Finally, our protocol is correct since:

$$\begin{aligned}
\texttt{Sharekey}(ID_A, pk^{(A)}, ID_B, sk_s^{(B)}) &= e(z_s^{(B)}, g_{\ell+1},\ldots,g_n, pk^{(A)})\\
&= e(h_{b_1}^{x_B}, h_{b_1b_2},\ldots,h_{b_1b_2\cdots b_\ell}, g_{\ell+1},\ldots,g_n, g^{x_A})\\
&= e(h_{b_1}^{x_A}, h_{b_1b_2},\ldots,h_{b_1b_2\cdots b_\ell}, g_{\ell+1},\ldots,g_n, g^{x_B})\\
&= e(z_s^{(A)}, g_{\ell+1},\ldots,g_n, pk^{(B)})\\
&= \texttt{Sharekey}(ID_B, pk^{(B)}, ID_A, sk_s^{(A)}).
\end{aligned}$$

Our Protocol in a Nutshell. The `Update` algorithm must be a one-way function to ensure the forward-security, but correctness requires keeping a relation between the secret key and the public key. Therefore we cannot use an arbitrary one-way function (such as, for example, hash functions). Our secret key sk_s at a time period $s = b_1\cdots b_\ell$ can be divided into two parts: the element z_s used to share a secret key at the current time period, and the other ones ($\cup_{i\in\mathcal{I}_s}\{z_{b_1\cdots b_{i-1}1}\}$) that will be used to update the key. Since, $\forall i\in\mathcal{I}_s$, the strings $b_1\cdots b_{i-1}1$ are not a prefix of a previous time period s^*, no one can compute z_{s^*} from sk_s. Moreover, multilinearity of the map implies that z_s is an element A^x (if pk is g^x) with $A\in\mathbb{G}_\ell$, which ensures correctness of our scheme. The use of different parameters $(g_1,\ldots,g_n)$ and h_s, for $s\in\mathcal{S}_n$, offers an efficient way to answer the oracle queries while being able to introduce the challenge values at a selected period, as shown in section 4.2.

For example, assume that $n = 3$ and $s = 01$, then we have $sk_{01} = \{z_1\}\cup\{z_{01}\}$. Since the length of $s = 01$ is $2 < n = 3$, updating the secret key to the next time period $s' = 010$ consists of replacing z_{01} by $z_{010}\leftarrow e(z_{01}, h_{010})$ and $z_{011}\leftarrow e(z_{01}, h_{011})$, so the new secret key $sk_{s'}$ is $\{z_1, z_{011}\}\cup\{z_{010}\}$ (see Figure 1). Now, s' has reached the maximum length $n = 3$ so the `Update` algorithm will simply delete the element z_{010} to output the secret key of the following time period 011.

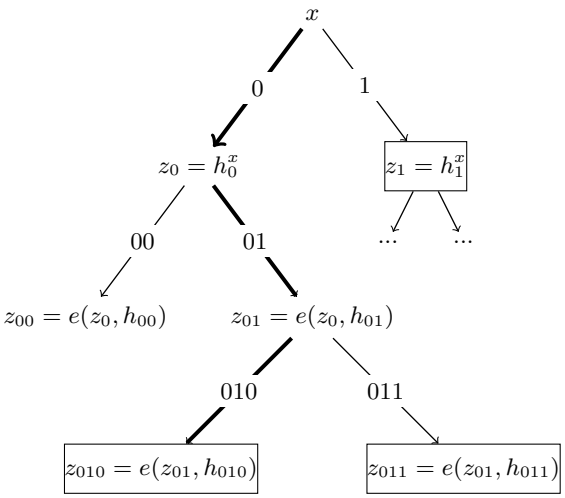

Fig. 1. Example for time period 010

4.2 Security Analysis

Theorem 1. *In the registered-key model, our forward-secure NIKE scheme with $2^{n+1}-2$ time periods is secure under the $(n+1)$-MDDH assumption.*

Proof. Let $\mathcal{A}$ be an adversary against our forward-secure NIKE scheme such that $\varepsilon = \Pr[\mathbf{Exp}^{\mathsf{fs}}_{\mathcal{A}}(k) = 1] - 1/2$, then we construct $\mathcal{B}$, an adversary against the $(n+1)$-MDDH problem, as follows:

- $\mathcal{B}$ first makes a guess for the target time period $s^* = b_1^* \cdots b_{\ell^*}^* \xleftarrow{\$} \mathcal{S}_n$, of length ℓ^*, and the two distinct honest users involved in the target session, $i_0, i_1 \xleftarrow{\$} \{1, \ldots, q_H\}$ where q_H is a bound on the number of $\mathcal{O}$RegHon-queries.
- On input an $(n+1)$-MDDH challenge $(g, g^{x_1}, \ldots, g^{x_{n+2}}, G)$, $\mathcal{B}$ generates $(m_1, \ldots, m_n) \xleftarrow{\$} \mathbb{Z}_p^n$ and $n_s \xleftarrow{\$} \mathbb{Z}_p$ for each $s \in \mathcal{S}_n$, then sets:
  - $\forall i \in \{1, \ldots, n\} \backslash \{\ell^*\}$, $g_i \leftarrow (g^{x_i})^{m_i}$
  - $g_{\ell^*} \leftarrow g^{m_{\ell^*}}$
  - $\forall i \in \{1, \ldots, \ell^*\}$, $h_{b_1^* \cdots b_i^*} \leftarrow g^{x_i \cdot n_{b_1^* \cdots b_i^*}}$
  - $\forall s \in \mathcal{S}_n \backslash \{b_1^*, b_1^* b_2^*, \ldots, b_1^* \cdots b_{\ell^*}^*\}$, $h_s \leftarrow g^{n_s}$

This way, only elements h_t such that t is a prefix of s^* will be challenge elements. This enables $\mathcal{B}$ to handle any $\mathcal{O}$Breakin-query on time periods later than s^*. Similarly, setting g_{ℓ^*} as a non-challenge element allows $\mathcal{B}$ to answer any $\mathcal{O}$Reveal-query for time periods other than s^*. Now, $\mathcal{B}$ runs $\mathcal{A}$ with the above parameters and answers the different queries as follows:

- $\mathcal{O}$RegHon(ID): Upon receiving an i-th register honest query, for a new identity ID, $\mathcal{B}$ acts as follows: If $i \neq i_0$ and $i \neq i_1$ then it runs the Keygen-algorithm on ID and returns the resulting public key to $\mathcal{A}$. Else, we have $i = i_b$ for $b \in \{0, 1\}$ and ID will now be denoted ID_b. $\mathcal{B}$ will act as if $sk_0^{(b)} = \{h_1^{x_{n+1+b}}, h_0^{x_{n+1+b}}\}$ and then set $pk^{(b)} = g^{x_{n+1+b}}$, from the challenge input.
- $\mathcal{O}$RegCor(ID, pk): Upon receiving a public key pk along with a new identity ID, $\mathcal{B}$ registers them. Since we first consider the *registered-key model*, we assume that $\mathcal{B}$ extracts the secret key $sk = x$ during the proof of knowledge of the secret key.
- $\mathcal{O}$Breakin(ID, s): If $ID \neq ID_b$ for $b \in \{0, 1\}$, then $\mathcal{B}$ returns $sk_s^{(ID)}$. Else, the behaviour of $\mathcal{B}$ depends on s. If $s \leq s^*$, then $\mathcal{B}$ aborts. Else, $\mathcal{B}$ parses s as $b_1 \cdots b_k$. The secret key of ID_b at this time period is $sk_s^{(b)} = \cup_{i \in \mathcal{I}_s} \{z^{(b)}_{b_1 \cdots b_{i-1}1}\} \cup \{z_s^{(b)}\}$. $\mathcal{B}$ proceeds as follows for each $i \in \mathcal{I}_s$:
  - If $i > \ell^*$, then $h_{b_1 \cdots b_{i-1}1} = g^{n_{b_1 \cdots b_{i-1}1}}$ and thus $z^{(b)}_{b_1 \cdots b_{i-1}1} = e(h_{b_1}^{x_{n+1+b}}, h_{b_1 b_2}, \ldots, h_{b_1 \cdots b_{i-1}1})$ which $\mathcal{B}$ can compute as:
  $$e(h_{b_1}, h_{b_1 b_2}, \ldots, (g^{x_{n+1+b}})^{n_{b_1 \cdots b_{i-1}1}})$$
  - If $i \leq \ell^*$, then $c_1 \cdots c_i := b_1 \cdots b_{i-1}1$ cannot be a prefix of s^*, otherwise we would have $s^* \geq b_1 \cdots b_{i-1}1 > s$ (since $b_i = 0$). So there is $j < i$ such that $b_j^* \neq c_j$ or $b_i^* = 0$ (which means that for $j = i$, $b_j^* \neq 1 = c_i$), implying that $\mathcal{B}$ is able to return $z^{(b)}_{b_1 \cdots b_{i-1}1} = z^{(b)}_{c_1 \cdots c_i} = e(h_{c_1}^{x_n+1+b}, h_{c_1 c_2}, \ldots, h_{c_1 c_2 \cdots c_i})$ by computing:
  $$e(h_{c_1}, h_{c_1 c_2}, \ldots, h_{c_1 \cdots c_{j-1}}, (g^{x_{n+1+b}})^{n_{c_1 \cdots c_j}}, h_{c_1 \cdots c_{j+1}}, \ldots, h_{c_1 c_2 \cdots c_i})$$
  which is a valid value since $h_{c_1 \cdots c_j} = g^{n_{c_1 \cdots c_j}}$.

  Similarly, $\mathcal{B}$ is able to return $z_s^{(b)}$, since s is not a prefix of s^*, and thus $h_s = g^{n_s}$, which means that $\mathcal{B}$ can answer every $\mathcal{O}$Breakin-query on ID_b for time periods $s > s^*$.
- $\mathcal{O}$Reveal(ID_A, ID_B, s): If at least one of the involved identity is honest and different from ID_0 and ID_1, then $\mathcal{B}$ runs the Sharekey algorithm. Else, we may assume (without loss of generality) that $ID_B = ID_b$ for some $b \in \{0, 1\}$. Since $\mathcal{B}$ is able to answer $\mathcal{O}$Breakin-queries involving ID_0 or ID_1 for time periods $s > s^*$, it is able to answer any $\mathcal{O}$Reveal-queries for these time periods. We then only consider time periods $s \leq s^*$:

- $s < s^*$. We distinguish the two following cases.
  - If there is $t \in \mathcal{S}_n$ such that t is a prefix of s but not of s^*, then h_t is not a challenge element so $\mathcal{B}$ can compute z_t (and so z_s) in the same way as in $\mathcal{O}$Breakin-queries, and then return the valid shared key.
  - Else, $s = b_1^* \cdots b_k^*$ is a prefix of s^*, then $\mathcal{B}$ returns the valid shared key by computing:

$$e(h_{b_1^*}, \ldots, h_{b_1^* \cdots b_k^*}, g_{k+1}, \ldots, g_{\ell^*-1}, (g^{x_{n+1+b}})^{m_{\ell^*}}, g_{\ell^*+1}, \ldots, g_n, pk^{(A)}).$$

- $s = s^*$. If $\{ID_A, ID_B\} = \{ID_0, ID_1\}$ then $\mathcal{B}$ aborts. Else, ID_A is a corrupted user, and then $\mathcal{B}$ uses the extracted key x_{ID_A} to return the shared key between ID_A and ID_b.

– At the challenge phase, $\mathcal{A}$ outputs two identities ID_A and ID_B along with a time period s. If $s \neq s^*$ or $\{ID_A, ID_B\} \neq \{ID_0, ID_1\}$ then $\mathcal{B}$ aborts. Else, it returns $G^{\prod_{i=1}^{\ell^*} n_{b_1 \cdots b_i} \prod_{i=\ell^*+1}^{n} m_i}$ which is a valid shared key between ID_0 and ID_1 if $G = e(g^{x_1}, \ldots, g^{x_{n+1}})^{x_{n+2}}$ and a random element from the shared key space otherwise. $\mathcal{B}$ is then able, using the bit returned by $\mathcal{A}$, to distinguish the $(n+1)$-MDDH problem with an advantage greater than $\varepsilon/(Tq_H^2)$.

□

We stress that in order to have the $(n+1)$-MDDH problem intractable, one needs to use an $(n+1)$-leveled multilinear map. With more levels, this problem because easy.

4.3 Dishonest-Key Registration Model

The simulator $\mathcal{B}$ above only needs the *registered-key model* to answer $\mathcal{O}$Reveal-queries involving a dishonest identity A and one of the target identity ID_b during the time period s^*. Indeed, the public elements $\{g_1, ..., g_n\}$ and $\{h_s\}_{s \in \mathcal{S}_n}$ are constructed in such a way that at least one of those involved in the shared keys does not depend on a challenge element when $s \neq s^*$. The simulator is then able to output $shk_{s \neq s^*}^{A,ID_b}$ by replacing this element by $(g^{x_{n+1+b}})^r$ where $r \in \{m_1, ..., m_n\} \cup \{n_s\}_{s \in \mathcal{S}_n}$. This is no longer true for the time period s^*, so $\mathcal{B}$ will use the secret key of A, extracted during the $\mathcal{O}$RegCor-queries, to output $shk_{s^*}^{A,ID_b}$.

In [FHKP13], the authors provide an efficient way to avoid the *registered-key model* by using a chameleon hash function [KR00] $\mathcal{H} : \{0,1\}^* \times \mathcal{R} \to \mathbb{Z}_p$, where $\mathcal{R}$ is the random space. The public parameters *params* now contain three additional elements u_0, u_1 and u_2 used to compute the public keys. Indeed, besides g^x, users now compute $t \leftarrow \mathcal{H}(g^x || ID, r)$ for some random $r \in \mathcal{R}$ and $z \leftarrow (u_0 u_1^t u_2^{t^2})^x$, and set their public key as (g^x, z, r). Before sharing a key shk, correctness of the public key must be checked, which is possible in an n-linear setting (as long as $n > 1$).

To handle $\mathcal{O}$Reveal-queries involving a dishonest user, the reduction $\mathcal{B}$ will construct the parameters u_i as follows:

– $\mathcal{B}$ first selects at random $m_0, m_1 \xleftarrow{\$} \{0,1\}$ and $v_0, v_1 \xleftarrow{\$} \mathcal{R}$;
– $\mathcal{B}$ computes $t_0 \leftarrow \mathcal{H}(m_0, v_0)$, $t_1 \leftarrow \mathcal{H}(m_1, v_1)$ and a polynomial $p(t) = p_0 + p_1 \cdot t + p_2 \cdot t^2$ of degree 2 whose roots are t_0 and t_1;
– Now let b_1 be the prefix of length 1 of s^* in the security proof, then $\mathcal{B}$ sets $u_i \leftarrow h_{b_1}^{p_i} g^{q_i}$, for $i = 0, 1, 2$, where q_0, q_1, q_2 are the coefficients of another random polynomial of degree 2.

To register ID_0 or ID_1, $\mathcal{B}$ will use the secret key of the chameleon hash function to get r_0 or r_1 such that $\mathcal{H}(g^{x_{n+1+b}} || ID_b, r_b) = t_b$ and outputs $(g^{x_{n+1+b}}, g^{x_{n+1+b} \cdot q(t_b)}, r_b)$ for $b \in \{0,1\}$. This is a valid public key because $p(t_b) = 0$.

To answer $\mathcal{O}$Reveal-queries involving a dishonest identity A during the time period s^*, correctness of the public key is first checked. So we may assume the public key $pk^{(A)}$ is well-formed: $pk^{(A)} = \{g^{x_A}, h_{b_1}^{p(t_A) \cdot x_A} g^{q(t_A) \cdot x_A}, r_A\}$, where $t_A \leftarrow \mathcal{H}(g^{x_A} || ID_A, r_A)$. Let $pk^{(A)}[i]$ be the i-th element of $pk^{(A)}$, $\mathcal{B}$ is able to recover $h_{b_1}^{x_A}$ by computing $(pk^{(A)}[2]/(pk^{(A)}[1])^{q(t_A)})^{1/p(t_A)}$. Since b_1 is a prefix of s^*, $\mathcal{B}$ is able to recover $z_{s^*}^{(A)} = e(h_{b_1}^{x_A}, \ldots, h_{s^*})$, which is the secret element from $sk_{s^*}^{(A)}$ used by A to compute the shared key at the time period s^*.

The reduction $\mathcal{B}$ thus no longer needs the *registered-key model* to handle $\mathcal{O}\texttt{Reveal}$-queries. The resulting protocol is then secure even considering the *dishonest-key registration model*. However, as explained below, we can only use this idea with generic leveled multilinear groups, and this is not yet possible with the concrete constructions proposed by [GGH13] and [CLT13].

4.4 Adjustments to Existing Multilinear Maps

We currently do not have a concrete construction for leveled multilinear maps: Garg, Gentry and Halevi [GGH13], followed by Coron, Lepoint and Tibouchi [CLT13], have unfortunately just proposed approximations of such maps. Nevertheless, the differences between their schemes and generic leveled multilinear maps imply some changes in our protocol. The main drawback of these modifications is that our protocol does no longer support the *dishonest-key registration model* and thus requires a proof of knowledge of the secret key during the registration phase. This is due to the fact that we cannot select elements from the exponent group (called level-zero encoding in their paper), but only sample random ones. This is problematic because, in the previous security proof, simulations of answers to $\mathcal{O}\texttt{Reveal}$-queries involving a corrupted identity were possible due to the elements u_0, u_1 and u_2, constructed using some specific exponents, which were the coefficients of the polynomial $p(t)$. Moreover, in that security proof, there is a need for inverting the exponent $p(t_A)$. Using the terminology of [GGH13] and [CLT13], this means that, knowing a level-zero encoding of some c we have to compute a level-zero encoding of c^{-1} which is not known as possible. We thus cannot achieve security in the *dishonest-key registration model* and then only consider the *registered-key model*. The resulting scheme is then similar to the one described at the beginning of this section but requires the following adjustments:

- In the `Keygen` algorithm, the element x will now be a sampled level-0 encoding. The public key pk and the secret key sk_0 contain level-1 encodings which must be re-randomized (see Section 2) using the Rerand algorithm.
- Similarly, in the `Update` algorithm, the new values z_s must be randomized to prevent recovery of secret keys of previous time periods.
- In the `Sharekey` algorithm, the secret key cannot be $e(z_s^{(B)}, g_{\ell+1}, \ldots, g_n, pk^{(A)})$ since this value depends on the randomness used for randomizing encodings during previous steps. We then run the $\mathsf{Extract}$ algorithm on this value and define the shared secret key shk_s^{AB} as the output.

This instantiation of our protocol illustrates that randomness is compatible with NIKE schemes as long as the output of the `Sharekey` algorithm does not depend on it.

4.5 System Parameters

Our protocol requires $(n+1)$-leveled multilinear groups to handle $T = 2^{n+1} - 2$ time periods. The set of public parameters, consisting of $T + n + 1$ elements from $\mathbb{G}_1$, can be shorten using a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ (for example we may set $h_s \leftarrow \mathcal{H}(s)$), however, the security proof will require to model $\mathcal{H}$ as a random oracle.

The size of the secret key sk_s depends on the time period s, however, it never contains more than $n+1$ elements since $\#sk_s \leq \#\mathcal{I}_s + 1 \leq n + 1$. Even if the number of levels only grows logarithmically *w.r.t.* the number of time periods, the dependence between the former and the parameters size remains a problem for the existing multilinear maps. In the next section we describe a generalization of our protocol which provides a trade-off between the number of levels and the size of the secret key.

5 A General Framework

A natural goal when designing a forward-secure NIKE is to make the parameters independent of the number of time periods. Compared to the protocol described in Section 3.3, our protocol has decreased the number of elements in the secret key but has increased the number of levels of the multilinear

maps. It is possible to go further with this trade-off and thus achieve a protocol with only one element in the secret key (but with a larger number of levels). We show in this section that each protocol of this paper is actually a variant of a general framework allowing us to choose the number of elements in the secret keys or the number of levels (but not both of them). The idea is somewhat similar with the generic construction described in [BM99, Section 2] but with two major differences: First, Bellare and Miner considered certification chains where the secret key s_i at some time period i was used to certify the public key p_{i+1} of a new secret/public keys pair (s_{i+1}, p_{i+1}). Signatures issued at the time period i must then contain the full certification chain, namely $((\sigma_i, p_i), \ldots, (\sigma_2, p_2))$ where σ_i is the certificate on p_i *w.r.t* p_{i-1}. Using a binary tree, they reached a logarithmic number of elements in messages and in storage. However, this idea is suitable for signature schemes but not for NIKE, one reason being that the new public keys p_i will remain unknown to other parties unless one publishes them all at the beginning, which would correspond to the trivial solution. The second difference is that, with their solution, a signing key (the secret key along with the stored values (σ_i, p_i)) with constant size is unachievable whereas this is theoretically possible with our construction (see Section 5.2).

5.1 The Framework

In the previous section, we considered the set $\mathcal{S}_n$ of bitstrings of size smaller than n. In this section we rather consider the sets $\mathcal{S}_n^{(m)} := \{b_1 \cdots b_k : k \leq n \text{ and } 0 \leq b_i < m\}$ for any integer $m > 0$ (the previous section therefore corresponds to the particular case where $m = 2$). Each string still refers to a time period, a protocol using $(n+1)$-leveled multilinear groups then ensures $\frac{m}{m-1} \times (m^n - 1)$ time periods if $m > 1$ and n time periods if $m = 1$[1].

- `Setup` and `Sharekey` algorithms are the same as the ones described in Section 4.1.
- `Keygen`(ID): The user ID first selects $x \xleftarrow{\$} \mathbb{Z}_p$ and then outputs $pk \leftarrow g^x$ and the secret key of ID at the first time period, $sk_0 \leftarrow \{h_{m-1}^x, \ldots, h_1^x, h_0^x\}$. In the following, for each $s = b_1 \cdots b_\ell \in \mathcal{S}_n^{(m)}$, $z_s^{(ID)}$ will still denote the following element of $\mathbb{G}_\ell$: $e(h_{b_1}^x, h_{b_1 b_2}, \ldots, h_{b_1 b_2 \cdots b_\ell})$.
- `Update`(sk_s): Let ℓ be the length of $s = b_1 \cdots b_\ell$ and $\mathcal{I}_s$ be the set $\{1 \leq i \leq \ell : b_i < m-1\}$. Then, $sk_s = \bigcup_{i \in \mathcal{I}_s} \bigcup_{b_i < b \leq m-1} \{z_{b_1 \cdots b_{i-1} b}\} \cup \{z_s\}$ and the algorithm proceeds as follows:
  - If $\ell < n$, then the next bitstring is $s||0$, the algorithm computes $z_{s||0} \leftarrow e(z_s, h_{s||0})$, $z_{s||1} \leftarrow e(z_s, h_{s||1}), \ldots, z_{s||m-1} \leftarrow e(z_s, h_{s||m-1})$, and returns $sk_{s||0} \leftarrow (sk_s \backslash \{z_s\}) \cup \{z_{s||m-1}, \ldots, z_{s||1}, z_{s||0}\}$.
  - If $\ell = n$, then we have $s = b_1 \cdots b_n$. If $b_i = m-1$, for all i, then we have reached the last time period and the algorithm returns $\perp$. Else, let j be the greatest integer such that $b_j < m-1$, the next time period s^* is then $b_1 \cdots b_{j-1}(b_j + 1)$. The algorithm then returns $sk_{s^*} \leftarrow \bigcup_{i \in \mathcal{I}_s, i \leq j} \bigcup_{b_i < b \leq m-1} \{z_{b_1 \cdots b_{i-1} b}\} \cup \{z_s\} \subset sk_s$.

The proof of correctness is similar to the case where $m = 2$: each time we move from a period s of length ℓ to a period $s||0$ of length $\ell + 1$, the `Update` algorithm simply stores the elements z_{s^*} for every strings s^* of length $\ell + 1$ whose prefix is s. Evolution of the secret key thus remains possible while recovery of elements z_t with $t \leq s$ is impossible. Adaptation of the security proof is straightforward.

5.2 System Parameters

The relation $T = \frac{m}{m-1} \cdot (m^n - 1)$ (or $T = n$ if $m = 1$) illustrates the trade-off between the number of levels $(n+1)$ of the multilinear map and the parameter m affecting the size of the secret key. Indeed, our algorithms `Keygen` and `Update` lead to a secret key containing up to $n(m-1)+1$ elements. Let us focus on the two following particular cases:

Case 1: $n = 1$. Our protocol only requires two levels from the multilinear map so we can use conventional bilinear groups. The secret key at the first time period is $\{h_{m-1}^x, \ldots, h_0^x, \}$ (with $m = T$), the update algorithm is to simply delete the last element of the secret key. This protocol is then exactly the same as the one described in Section 3.3.

[1] This comes from the fact that $p_n = \#\mathcal{S}_n^{(m)}$ follows an arithmetico-geometric progression $p_n = m(p_{n-1} + 1)$, with $p_0 = 0$.

Case 2: $m = 1$. For convenience, we will denote in such a case the time periods by $\{1, ..., T\}$ rather than $\{0, 00, \ldots, 00\cdots000\}$. The secret key always contains one element $z_i = e(h_1^x, h_2, ..., h_i) \in \mathbb{G}_i$, updating it just consists in computing $z_{i+1} \leftarrow e(z_i, h_{i+1})$. Assuming, as Papamanthou *et al* [PTT10], the existence of multilinear maps where the size of the different groups $\mathbb{G}_i$ is independent of the number of levels leads to a protocol where the sizes of the parameters and of the secret and public keys are independent of the number of time periods T. However, implementations of such a protocol with the maps from [GGH13] and [CLT13] will not achieve constant size since the size of the elements in the secret key will actually depend on the number of levels and so on the number of time periods.

Trade-off. Between these two extreme cases, one can choose suitable parameters according to the performance of the selected multilinear map. The maximal number of elements in the secret key is $N = n(m-1)+1$, and can be expressed as a function of m and T using the relation $T = \frac{m}{m-1} \cdot (m^n - 1)$ if $m > 1$. We then get:

$$N = 1 + \frac{m-1}{\log(m)} \cdot \log\left(\frac{m-1}{m} \times T + 1\right)$$

which is an increasing function of m in $]1; T]$. There is thus no *optimal* choice for the parameters (m, n). They have thus to be chosen according to some external constraints.

6 Conclusion

In this paper, we have first proposed two new security models for forward-secure non-interactive key exchange scheme, in order to limit damages in case of key exposure. In the first *registered-key* model, the certificate authority is assumed to strictly check the knowledge of the secret keys before certifying public keys together with an identity, whereas in the second *dishonest-key registration* model, the certificate authority just checks the identity but not the knowledge of the secret key associated to the public key. The latter model encompasses related-key attacks, where an adversary would try to generate public keys related to honest-user keys.

We then have proposed a construction that can be secure in the strongest security model using generic leveled multilinear maps. Unfortunately, concrete multilinear maps do not yet satisfy all the required properties, and thus our concrete construction just provides forward security in the registered-key model. We have thus pointed out a gap between generic leveled multilinear maps and concrete ones.

Of course, the efficiency of our construction depends to a large extent on the one of such maps. However, our construction can be made practical by tuning the number of levels of the multilinear map, impacting the size of the secret key but not the number of time periods.

Acknowledgments

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

[ABP13] Michel Abdalla, Fabrice Ben Hamouda, and David Pointcheval. Tighter reductions for forward-secure signature schemes. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 292–311. Springer, February / March 2013.

[And97] Ross Anderson. Two remarks on public key cryptology., 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. Springer, December 2000.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, April 2006.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, August 1999.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. In *Contemporary Mathematics*, volume 324, pages 71–90, 2003.

[BSSW06] Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters. Forward-secure signatures with untrusted update. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 191–200. ACM Press, October / November 2006.

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer, April 2008.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer, August 2013.

[DE06] Régis Dupont and Andreas Enge. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics*, 154(2):270–276, 2006.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer, February / March 2013.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 513–530. Springer, August 2013.

[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, August 1986.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, May 2013.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 494–512. Springer, August 2013.

[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, August 2001.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *ISOC Network and Distributed System Security Symposium – NDSS 2000*. The Internet Society, February 2000.

[KR02] Anton Kozlov and Leonid Reyzin. Forward-secure signatures with fast key update. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 241–256. Springer, September 2002.

[PS09] Kenneth G. Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptography*, 52(2):219–241, 2009.

[PTT10] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal authenticated data structures with multilinear forms. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *PAIRING 2010: 4th International Conference on Pairing-based Cryptography*, volume 6487 of *Lecture Notes in Computer Science*, pages 246–264. Springer, December 2010.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing over elliptic curve. *Symposium on Cryptography and Information Security*, 2000.