Efficient key exchange protocols for wireless networks and mobile devices

Katrin Hoeper and Guang Gong
khoeper@engmail.uwaterloo.ca, ggong@calliope.uwaterloo.ca
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, N2L 3G1, Canada

Abstract

In this paper, we design efficient authenticated key exchange protocols that take the special constraints of wireless networks and mobile devices, such as limitations in bandwidth, computational power, memory space and battery lifetime, into account. To accomplish the design goals, we show how a Diffie-Hellman (DH) key exchange can be securely integrated into a digital signature scheme (DSS) using either LUC, XTR or GH as crypto schemes to enable an authenticated key agreement between two parties. The integration, as first proposed by Arazi, saves one costly computation step and significantly reduces the bandwidth. We prove that our protocols resist the known-key attack that Nyberg and Rueppel presented on Arazi's scheme. We demonstrate that our integrated protocols are more efficient in terms of bandwidth and computational costs than combined DH-DSS protocols by providing a theoretical analysis and experimental results. Our integrated protocols are on average 20 % more efficient in computational time and up to 30 % more efficient in terms of bandwidth than a combined protocol and thus well-suited for an implementation in mobile wireless networks.

Keywords: `authentication, key agreement, digital signature scheme, Diffie-Hellman key exchange, known-key attack, LUC, GH, XTR`

1 Introduction

Digital communication has found its way into nearly all parts of our daily life. A rapidly growing number of such applications use wireless communication channels and mobile devices. All communication channels, and especially the wireless ones, require protection from eavesdropping and other malicious attacks. Mutual authentication and establishment of a session key are both essential steps to bootstrap secure communication channels. However, wireless channels are constrained in their bandwidth and most mobile devices are constrained in their computational power, memory space and battery lifetime. For these reasons, we cannot simply take existing protocols that were originally designed for wired networks and more powerful devices and apply them to wireless mobile networks. In order to cope with the new challenges, we need to design new protocols that take the special requirements and limitations of wireless networks and mobile devices into account. Hence, we are seeking protocols that are efficient in terms of (1) bandwidth, to address the limitations of wireless communications channels and the restricted power sources of mobile devices; and (2)

computational performance, to address the limitations of computing and battery power of mobile devices.

In this work, we consider protocols that provide mutual authentication and authenticated key exchange by combining a Diffie-Hellman (DH) key exchange and a digital signature scheme (DSS). In this kind of *DH-DSS* protocols ephemeral public keys are first signed using a DSS for authentication purposes and then used in a DH key agreement to derive a fresh session key. As a step towards an efficient protocol design, we show how a DH key exchange can be securely integrated into a DSS using LUC [15, 6, 16], XTR [11, 17] and GH [8, 7] as cryptographic primitives. We refer to these protocols as *integrated DH-DSS* protocols, whereas we refer to the ordinary combined ones as *combined DH-DSS* protocols. Furthermore, we refer to the used crypto primitives as second-order or third-order linear feedback shift register (LFSR)-based crypto schemes for the remainder of the paper. The integration step in the proposed protocols saves one heavy computation step and significantly reduces the required bandwidth. Besides the advantage due to the integration, the underlying crypto primitives used in all proposed protocols in this work help to further increase the performance because the primitives can be efficiently implemented in hardware and use smaller fields for computations than crypto schemes based on the exponential function, such as the DSA. For all the discussed reasons the proposed integrated DH-DSS protocols are attractive for an implementation in wireless mobile networks.

The idea of integrating a DH key exchange into a DSS was first proposed by Arazi [1] and later shown to be vulnerable to a known-key attack by Nyberg and Rueppel [13]. In this paper we analyze the known-key attack and formalize conditions that underlying crypto systems need to satisfy in order for the attack to work. We demonstrate that those conditions are satisfied in a finite field and elliptic curve implementation of Arazi's protocol. Next we introduce integrated DH-DSS protocols that are based on LUC, XTR and GH, respectively, and show that those protocols are immune to the known-key attack. After showing that the presented protocols resist the known-key attack we analyze their performance. In our performance analysis we analyze the advantage of integrated protocols over combined DH-DSS protocols with respect to computational complexity and required bandwidth. Both, our analytical and experimental results show that the integrated protocols are about 20% more efficient in terms of computational performance. The analytical analysis shows that the bandwidth is up to 30% smaller in the integrated protocols. In addition, we show that the approach of integrating DH key exchange into a DSS is applicable to all DH-DSS authentication and authenticated key exchange protocols to significantly improve their efficiency while maintaining the original security level. We discuss the approach for general protocols and demonstrate it for the Internet Key Exchange (IKE) protocol [9] where the integrated IKE protocols is more efficient than the original protocol while maintaining all its security properties. Thus, we developed a tool to transform existing secure protocols into efficient secure protocols that are well-suited for wireless networks and constrained network devices.

The rest of the paper is organized as follows. In the next section, we briefly review Arazi's schemes and analyze the conditions for a successful known-key attack on the scheme. In Section 3, we introduce LFSR-based integrated protocols for the quadratic and cubic cases. Then we show in Section 4 that the presented protocols are immune to the known-key attack. In Section 5, we analyze and compare the computational and communications performance of the presented protocols and combined DH-DSS protocols. We fortify these observations with our experimental results. Finally, we draw some conclusions in Section 7.

2 Arazi's Scheme and the Known-Key Attack

In this section, we first briefly review Arazi's integrated DH-DSS protocol and Nyberg's and Rueppel's known-key attack on it. Next, we formalize under which conditions a known-key attack is successful and show that those conditions are satisfied in a finite field and elliptic curve implementations of Arazi's protocol. Finally, we discuss some related work and point out how our work differs from it.

2.1 Arazi's Scheme

System parameters: In the following section, p is a large prime, q is a prime divisor of $p-1$, g is an element of multiplicative order q in $\mathbb{Z}_p$ and $h(\cdot)$ is a hash function.
User parameters: A party A has a long-term private key $x_A \in \mathbb{Z}_q$ and a long-term public key $y_A = g^{x_A} \pmod p$.

Protocol 1. Arazi's integrated DH-DSS protocol

PROTOCOL MESSAGES:

1. $A \longrightarrow B : t_A, m_A$
2. $A \longleftarrow B : t_B, m_B$

SHARED KEY: $K = m_A^{k_B} = m_B^{k_A} = g^{k_A k_B} \pmod p$.

Protocol steps: We consider an execution of Arazi's protocol between two parties A and B. We assume that both parties have authentically exchanged their long-term public keys y_A and y_B prior to the protocol execution, for example, by using public key certificates. The protocol flow is illustrated in Protocol 1 and the computations in each step are as follows. Party A randomly chooses an ephemeral secret key $k_A \in \mathbb{Z}_q$ and derives an ephemeral public key $m_A = g^{k_A} \pmod p$. A's signature (r_A, t_A) on m_A is derived by a simple modulo operation, $r_A = m_A \pmod q$, and solving the signing equation $h(m_A) \equiv k_A t_A - x_A r_A \pmod q$ in t_A. Party B performs the symmetric steps. Upon signing the ephemeral public keys, A sends (t_A, m_A) to B and B sends (t_B, m_B) to A. Each party derives the second part r of the signature by modulo q reduction of the received message m. Now both parties verify the received signature according to the DSA. If both signatures are valid the two parties A and B can compute the shared key K according to the DH key agreement. In Arazi's scheme DSA and DH can be replaced by ECDSA and ECDH, respectively. Note that computing the signature part r requires an additional computation step for each party in a combined DH-DSS protocol, because both parties need to choose another random number k_S and compute $r = g^{k_S} \pmod p$.

2.2 Known-Key Attack

Analyzing the protocol flows and computation steps of Protocol 1 for a DH and DSA implementation, we can set up the following equations. By multiplying the two signature parts t_A and t_B and the ephemeral secret keys k_A and k_B and using the result as the exponent of g, we get Equation (1) below. From this equation it is trivial to derive Equation (2) as follows.

$$g^{t_A t_B k_A k_B} = g^{h(m_A)h(m_B) + x_A x_B r_A r_B + x_A r_A h(m_B) + x_B r_B h(m_A)} \quad (1)$$

$$= g^{h(m_A)h(m_B)} g^{x_A x_B r_A r_B} g^{x_A r_A h(m_B)} g^{x_B r_B h(m_A)} \quad (2)$$

If we now replace $g^{t_A t_B k_A k_B}$ by $K^{t_A t_B}$ on the left hand side and $g^{x_A r_A h(m_B)}$ by $y_A^{r_A h(m_B)}$ and $g^{x_B r_B h(m_A)}$ by $y_B^{r_B h(m_A)}$ on the right hand side of Equation (2) we obtain

$$K^{t_A t_B} = g^{h(m_A)h(m_B)}(g^{x_A x_B})^{r_A r_B} y_A^{r_A h(m_B)} y_B^{r_B h(m_A)}. \quad (3)$$

In [13], the authors discovered that the only parameters that are not publicly known in Equation (3) are K and $g^{x_A x_B}$. Consequently, if an attacker knows a session key K, he can obtain $g^{x_A x_B}$ and thus obtain any other session key shared between the same two parties. This follows that Arazi's scheme suffers a known-key attack and should thus not be implemented. As previously mentioned, in the protocol DSA and DH can be replaced by ECDSA and ECDH, respectively. For an elliptic curve based implementation we can derive an equation similar to Equation (3), i.e. an equation in which the session key K and a static long-term secret key are the only unknown. Consequently, an ECDH-ECDSA implementation of Protocol 1 also suffers the known-key attack.

We now have a closer look why the known-key attack works. For a successful attack on Arazi's integrated DH-DSS protocol the underlying crypto scheme must satisfy two conditions:
Condition 1. Equation (2) can be directly derived from Equation (1).
Condition 2. The left and right hand side of Equation (2) can be replaced by the session key and the long term public keys to obtain an equation similar to (3), i.e. an equation in which the session key and a static secret long-term key are the only unknown. The equation must be solvable in both unknown values, i.e. the static secret key value can be computed once one session key K is known and upon deriving this value the equation can be solved in K to determine other session keys.

A known-key attack can only be launched if and only if Conditions 1 and 2 are both satisfied. Both, DSA and ECDSA satisfy Condition 1 and 2, i.e. we can derive Equation (2) from Equation (1) and finally obtain Equation (3). We will now demonstrate that DSA and ECDSA both satisfy Condition 1 because they are homomorphic schemes. Let α be an element in $GF(Q)^*$ of order N, where Q is a prime or a power of a prime. The group $\mathbb{G} = \{\alpha^i \mid i = 0, 1, \ldots, N-1\}$ is a subgroup of $GF(Q)^*$ under multiplication. The map $\tau : k \mapsto \alpha^k$ is an isomorphism from the additive group $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ to $\mathbb{G}$. In other words, τ is an one-to-one map from $\mathbb{Z}_N$ to $\mathbb{G}$, and τ satisfies the homomorphic condition

$$\tau(n+m) = \tau(n)\tau(m), \text{for all } n,\, m \in \mathbb{Z}_N \quad (4)$$

i.e., $\alpha^{n+m} = \alpha^n \alpha^m$ for all $n, m \in \mathbb{Z}_N$. Note that in the elliptic curve case, α is replaced by a point on an elliptic curve with order N, and $\mathbb{G}$ is written as an additive group. Hence, both DSA and ECDSA satisfy the homomorphic condition (4) which directly follows that Condition 1 is satisfied.

It is easy to see that DSA satisfies Condition 2, because Equation (3) can be solved in the session key K and the static secret key $g^{x_a x_B}$. The same argument applies to the elliptic curve-based implementation of the protocol.

2.3 Related Work

In addition to the known-key attack, a small subgroup attack has been proposed on Arazi's scheme in [2]. However, this attack can easily be prevented by validating the order of the received public key and thus will no longer be considered as a weakness for the remainder of this paper.

In [10], the authors introduced three protocols that integrate the DH key exchange into the DSA using Arazi's original idea. Like the protocols that we propose in this paper, these protocols also resist the known-key attack. In addition, the two-round protocol in [10] has a similar computational

complexity than our integrated DH-DSS protocol. However, none of the protocols proposed in [10] provide key control, i.e. the shared keys in these protocols are not computed in a contributory fashion. One part of the session key is the long-term public key of the receiver, i.e. a static key that is not freshly chosen in each run. In Protocol 1 in [10], only party A determines the key while in protocols 2 and 3, the two shared keys are derived each by one party alone and are used for only one communication direction. The property of key control is desirable in key exchange protocols because it prevents that a single principal controls the value of the shared key. If key control is provided a principal can ensure that the generated session key is fresh and has good random properties. Furthermore, it is desirable to have only one session key that can be used for both communication directions. The integrated DH-DSS protocols presented in this paper all provide key control and have the advantage that the same session key K can be used for both communications directions. Furthermore, the presented protocols are based on different cryptographic primitives as the ones in [10].

3 LFSR-Based Integrated DH-DSS Protocols

In this section, we present LFSR-based integrated DH-DSS protocols. The protocols can be implemented for any order n of the LFSRs but we limit our discussion to the quadratic and cubic cases. We consider one example for a quadratic protocol implementation based on the LUC scheme and two examples for cubic implementations based on the XTR and GH schemes, respectively. The protocol flow is similar to Protocol 1 and given in Protocol 2, where the ephemeral public keys are sequence terms f_k. The protocol messages in Protocol 2 are for general order n, whereas the session key computation is specific to LUC, XTR and GH. The protocol steps for quadratic and cubic implementations are described in great detail in the following two sections. We omit a general introduction to LFSR-based crypto schemes and the reader is referred to [5] for a general description, to [15, 6, 16] for a detailed description of the LUC scheme, to [11, 17] for XTR, and to [8, 7] for the GH scheme.

Protocol 2. LFSR-based integrated DH-DSS protocol

PROTOCOL MESSAGES:

1. $A \longrightarrow B : t_A, f_{k_A}$
2. $A \longleftarrow B : t_B, f_{k_B}$

SHARED KEY: $K = s_{k_A}(f_{k_B}(x)) = s_{k_B}(f_{k_A}(x)) = s_{k_A k_B}$ for LUC and XTR and $K_1 = K$, $K_2 = s_{-k_A k_B}$ for GH.

3.1 Quadratic LFSR-Based Integrated DH-DSS Protocols

The LUC scheme employs a second-order characteristic sequence and can be used for an efficient implementation of a quadratic LFSR-based integrated DH-DSS protocol.

Preliminaries: Let $\{s_i\}$ be a second-order characteristic sequence over a finite field $GF(Q)$, generated by $f(x) = x^2 - ax + 1$, $a \in GF(Q)$, which is a irreducible polynomials, i.e.,

$$s_{i+2} = as_{i+1} - s_i \text{ for all } i \geq 0, \text{ where } (s_0, s_1) = (2, a). \tag{5}$$

For convenience, we sometimes use $s_i(f(x))$ to represent the sequence $\{s_i\}$ which is generated by $f(x)$. The protocol requires an algorithm to compute the sequence term s_k for a given k such as given in [15, 16]. Another algorithm is required in the verification procedure to compute a mixed term s_{v+x} for a given v and unknown x, e.g. in [16].

System public parameters: p is a prime and $Q = p$. Choose $f(x) = x^2 - ax + 1$ which is irreducible over $GF(Q)$ with period q, where q is a prime factor of $p+1$ and $h(\cdot)$ is a hash function.

User parameter: A party A chooses x_A, with $0 < x_A < q$ as its private key and computes y_A as its public key, with $y_A = (s_{x_A}, s_{x_A+1})$.

Sign the ephemeral public keys: Party A signs its ephemeral public key f_{k_A}. Therefore, A performs the following steps:

QS-1 Randomly chooses a k_A, with $0 < k_A < q$, which A uses as its ephemeral secret key for one session.

QS-2 Computes the ephemeral public key $f_{k_A} = s_{k_A}$ and the hash value $h_A = h(s_{k_A})$.

QS-3 Sets $r_A = s_{k_A}$ and solves the signing equation: $t_A \equiv k_A^{-1}(h_A + x_A r_A) \pmod{q}$.

QS-4 Sends (t_A, f_{k_A}) to party B.

Party B performs the symmetric operations to sign its ephemeral public key f_{k_B}.

Verify the ephemeral public keys: Upon receiving the message from A, B verifies the signature on A's ephemeral public key by performing the following steps:

QV-1 Sets $v \equiv h_A r_A^{-1} \pmod{q}$ and $u \equiv r_A^{-1} t_A \pmod{q}$.

QV-2 Computes $T_1 = s_{v+x_A}$ from y_A.

QV-3 Computes $T_2 = s_u(f_{k_A}(x))$, the u-th term of the characteristic sequence $f_{k_A}(x) = x^2 - s_{k_A}x + 1$.

QV-4 Checks whether

$$T_1 = T_2. \tag{6}$$

If (6) is true, B accepts. Otherwise, B rejects.

A has to perform the symmetric operations to verify B's ephemeral public key.

Session key computation: Upon the successful verification of the ephemeral public keys, both parties are able to compute the common key K as shown below. The key is then used to derive a session key in a predefined fashion.

QSK-1 $K = s_{k_A}(f_{k_B}(x)) = s_{k_B}(f_{k_A}(x)) = s_{k_A k_B}$.

3.2 Cubic LFSR-Based Integrated DH-DSS Protocols

In this section we consider two cubic schemes, the GH and the XTR scheme. Both schemes employ a third-order characteristic sequence and can be used to efficiently implement integrated DH-DSS protocols.

Preliminaries: Let $\{s_i\}$ be a third-order characteristic sequence over a finite field $GF(Q)$, generated by $f(x) = x^3 - ax^2 + bx - 1$, $a, b \in GF(Q)$, which is an irreducible polynomials, i.e.,

$$s_{i+3} = as_{i+2} - bs_{i+1} + s_i \quad \text{for all } i \geq 0, \quad \text{where } (s_0, s_1, s_2) = (3, a, a^2 - 2b). \tag{7}$$

As for the LUC scheme, implementing GH and XTR requires the computation of a sequence term s_k for given k and an algorithm to compute a mixed sequence term s_{v+x}. The reader is referred to [11] for efficient algorithms in the XTR scheme and to [7, 8] for the GH scheme.

System public parameters: p is a prime, $Q = p$ for GH and $Q = p^2$ for XTR. Choose $f(x) = x^3 - ax^2 + bx - 1$ for both GH and XTR, where $b = a^p$ in the XTR case. The polynomial $f(x)$ is irreducible over $GF(Q)$ with period q, where q is a prime factor of $p^2 + p + 1$ for GH, and a prime factor of $p^2 - p + 1$ for XTR. $h(\cdot)$ is a hash function.

User parameter: A party A chooses x_A, with $0 < x_A < q$ as its private key and computes y_A as its public key, with $y_A = (s_{x_A}, s_{x_A+1}, s_{x_A+2})$.

Sign the ephemeral public keys: Party A wants to sign its ephemeral public key f_{k_A} and performs the following steps:

CS-1 Randomly chooses a k_A, with $0 < k_A < q$, which A uses as ephemeral secret key for one session.

CS-2 Computes the ephemeral public key f_{k_A} and its hash h_A, where $f_{k_A} = s_{k_A}$ and $h_A = h(s_{k_A})$ for XTR, and $f_{k_A} = (s_{k_A}, s_{-k_A})$ and $h_A = h(s_{k_A} + s_{-k_A})$ for GH, respectively.

CS-3 Sets $r_A = s_{k_A}$ and solves the signing equation: $t_A \equiv k_A^{-1}(h_A + x_A r_A) \pmod{q}$.

CS-4 Sends (t_A, f_{k_A}) to B.

B performs the symmetric operations to sign its ephemeral public key.

Verify the ephemeral public keys: Upon receiving the message from A, B verifies the signature on A's ephemeral public key by performing the following steps:

CV-1 Sets $v \equiv h_A r_A^{-1} \pmod{q}$ and $u \equiv r_A^{-1} t_A \pmod{q}$.

CV-2 Computes $T_1 = s_{v+x_A}$ from y_A.

CV-3 Computes $T_2 = s_u(f_{k_A}(x))$, the u-th term of the characteristic sequence $f_{k_A}(x) = x^3 - s_{k_A}x^2 + s_{-k_A}x - 1$, with $s_{-k_A} = s_{k_A}^p$ in the XTR case.

CV-4 Checks whether

$$T_1 = T_2. \tag{8}$$

If (8) is true, B accepts. Otherwise, B rejects.

A has to perform the symmetric operations to verify B's ephemeral public key.
Session key computation: Upon the successful verification of the ephemeral public keys, both parties are able to compute the common key K as shown below

CSK-1 $K = s_{k_A}(f_{k_B}(x)) = s_{k_B}(f_{k_A}(x)) = s_{k_A k_B}$ for XTR

CSK-2 $K = (K_1, K_2)$ with $K_1 = s_{k_A}(f_{k_B}(x)) = s_{k_B}(f_{k_A}(x)) = s_{k_A k_B}$ and $K_2 = s_{-k_A}(f_{k_B}(x)) = s_{-k_B}(f_{k_A}(x)) = s_{-k_A k_B}$ for GH.

4 Immunity to Known-Key Attack of LFSR-Based Integrated DH-DSS Protocols

As discussed in Section 2.2, only integrated DH-DSS protocols based on crypto schemes satisfying both Conditions 1 and 2 are vulnerable to the known-key attack. We will now show that the quadratic and cubic LFSR-based integrated DH-DSS protocols presented in Section 3.1 and 3.2, respectively, resist the known-key attack. The presented security discussion in this section can be easily extended to LFSR-based schemes of general order n and it can be shown that all LFSR-based integrated DH-DSS protocols are immune to the known-key attack.

We first show that LUC, GH, and XTR-based integrated protocols do not satisfy Condition 1. Therefore, we form Equation (9) similar to Equation (1), i.e.

$$s_{k_A k_B t_A t_B} = s_{h_A h_B + h_A x_B r_B + h_B x_A r_A + x_A x_B r_A r_B} \tag{9}$$

In order to derive an equation similar to Equation (2) we would need to split the sequence term on the right hand side of Equation (9) into four separate sequence terms. In other words, LFSR-based crypto systems would need to have a property similar to the homomorphic property of DSA and ECDSA (Equation (4)). We introduce such a property in the following. Let ρ be the map which maps the index i to the sequence term s_i, i.e. $\rho : i \mapsto s_i, \quad 0 \leq i \leq N$. So, ρ is a map from the additive group $\mathbb{Z}_N$ to a subset of a multiplicative group of $GF(Q)$. The analogue case of the homomorphic property as given in Equation (4) for a sequence $\{s_i\}$ is given by

$$\rho(n+m) = \rho(n)\rho(m) \tag{10}$$

i.e., $s_{n+m} = s_n s_m$ which means the product of the n-th and the m-th terms is equal to the $(n+m)$-th term of the sequence. We will refer to (10) as the multiplicable property of a sequence in the paper. Otherwise, i.e. if Equation (10) is not satisfied, the sequence is non-multiplicable.

Generally LFSR-based schemes are non-multiplicable, which we show in the following. From [6] and [8] respectively, for any n and m, we have

$$s_{n+m} = s_n s_m - s_{n-m} \tag{11}$$

if $\{s_k\}$ is a second-order characteristic sequence given by (5), and

$$s_{n+m} = s_n s_m - s_{n-m} s_{-m} + s_{n-2m} \tag{12}$$

if $\{s_k\}$ is a third-order characteristic sequence given by (7). Thus, for randomly chosen integers n and m, $s_{n+m} \neq s_n s_m$ holds with probability close to 1, i.e. $\rho(n+m) \neq \rho(n)\rho(m)$. The considered

quadratic or cubic LFSR-based schemes are multiplicable if $s_{n+m} = s_n s_m$ in Equation 11 and 12, respectively, which is true if and only if

$$\begin{aligned} s_{n-m} &= 0 \quad \text{in the quadratic case} && (13) \\ s_{n-2m} - s_{n-m}s_{-m} &= 0 \quad \text{in the cubic case.} && (14) \end{aligned}$$

Condition 1 holds if and only if Equation (13) is true in the quadratic case and Equation (14) is true in the cubic case for all four sequence terms $s_{h_A h_B}$, $s_{h_A x_B r_B}$, $s_{h_B x_A r_A}$ and $s_{x_A x_B r_A r_B}$, respectively. Note that the probability that Equation 10 holds is very small and thus the probability that it holds for four different sequence terms is negligibly small.

Despite the small odds, we now assume that Condition 1 holds and analyze if Condition 2 is satisfied in the considered quadratic and cubic LFSR-based schemes. Given Condition 1 holds, we can derive the following equation

$$s_{k_A k_B t_A t_B} = s_{h_A h_B} s_{h_A x_B r_B} s_{h_B x_A r_A} s_{x_A x_B r_A r_B} \tag{15}$$

We can now derive an equation similar to Equation (3) by replacing sequence terms with publicly known sequence terms in Equation (15) until only two unknown parameters are left, namely the session key $s_{k_A k_B}$ and the static long-term secret key $s_{x_A x_B}$. Doing so we obtain Equation (16) below.

$$s_{t_A t_B}(f_{k_A k_B}) = s_{h_A h_B} s_{h_A r_B}(f_{x_B}) s_{h_B r_A}(f_{x_A}) s_{r_A r_B}(f_{x_A x_B}) \tag{16}$$

If one session key $s_{k_A k_B}$ (or ($s_{k_A k_B}$, $s_{-k_A k_B}$) in the GH case) is known, then $f_{k_A k_B}(x) = x^2 - s_{k_A k_B} x + 1$ or $f_{k_A k_B}(x) = x^3 - s_{k_A k_B} x^2 + s_{-k_A k_B} x - 1$ is known in the quadratic or cubic case, respectively. From (16), we can obtain the value for $s_{r_A r_B}(f_{x_A x_B})$. We set this value as c, i.e.

$$s_{r_A r_B}(f_{x_A x_B}) = c \tag{17}$$

or equally $s_{r_A r_B x_A x_B} = c$. For convenience, we set $e = r_A r_B$ and $d = x_A x_B$. Note that $s_{ed} = c$, which we may write as

$$s_d(f_e) = c \tag{18}$$

where $f_e(x) = x^2 - s_e x + 1$ in the quadratic and $f_e(x) = x^3 - s_e x^2 + s_{-e} x - 1$ in the cubic case and s_e ((s_e, s_{-e}) in the GH case) is known. However, obtaining d from (18) is a discrete logarithm problem of the original quadratic or cubic LFSR-based schemes. So we only need to consider (17), i.e. $s_e(f_d) = c$ where both c and e are known, and the characteristic polynomial $f_d(x) = x^2 - s_d x + 1$ or $f_d(x) = x^3 - s_d x^2 + s_{-d} x - 1$ is unknown. According to the general theory of recursive sequences [5], specifically, according to the Berlekamp-Massey algorithm [14] together with the fact that the constant term in $f_d(x)$ is 1, we need $2n - 1$ consecutive sequence terms to recover $f_d(x)$ where n is the degree of $f_d(x)$. In other words, we need to recover s_d (or (s_d, s_{-d}) in the GH case) from the following equations:

$$s_{k+2}(f_d) - s_d s_{k+1}(f_d) + s_k(f_d) = 0 \text{ in } GF(Q), k \in \{e-2, e-1, e\} \tag{19}$$

for the quadratic case or

	Integrated DH-DSS	Combined DH-DSS
Generating ephemeral public key	1 po	1 po
Signing	-	1 po
Verifying	2 po	2 po
Computing shared key K	1 po	1 po
Total cost/party	4 po	5 po
Total cost/protocol execution	8 po	10 po

Table 1: Comparison of computational costs of integrated DH-DSS and combined DH-DSS protocols.

$$s_{k+3}(f_d) - s_d s_{k+2}(f_d) - s_{-d} s_{k+1}(f_d) + s_k(f_d) = 0 \text{ in } GF(Q), \quad k \in \{e-3, e-2, e-1, e\} \tag{20}$$

for the cubic case. For either (19) or (20), we only know $s_e(f_d)$. Thus from known e and $s_e(f_d)$ to recover f_d is infeasible. Hence, Condition 2 is not satisfied and whence it follows that the known-key attack will not work on the presented quadratic and cubic LFSR-based integrated protocols. We conclude that the considered LFSR-based schemes can be used to securely implement integrated DH-DSS protocols.

5 Performance Analysis

We first analyze and compare the computational cost and bandwidth requirements of the combined and integrated DH-DSS protocols. We then present our experimental results and show that the analytical observations were indeed consistent.

5.1 Theoretical Analysis of Computational and Communications Costs

The advantage in computational costs of the integrated DH-DSS over a combined DH-DSS is summarized in Table 1. In this table, we consider only the most expensive operations and refer to them as primitive operations (po), i.e., we count the number of modular exponentiations in a finite field implementation, such as DH-DSA, the scalar multiplications of points in elliptic curve schemes, such as ECDH-ECDSA, and the sequence term calculations in LFSR-based implementations, such as quadratic and cubic DH-DSA. The sequence term calculations are analogue operations to exponentiations in the DSA, and scalar multiplications for ECC. In an integrated protocol the ephemeral public key of each party is used as part of the signature equation, from which follows that each party can save one primitive operation. Thus, signing a message does not require any primitive operations and is virtually for free. We can see in Table 1 that the total cost per party is 4 po in the integrated protocol and 5 po in the combined one. Hence, we can expect a 20 % improvement in computational performance by implementing the integrated protocol instead of the combined one.

In addition to an improved computational performance, the communications performance of the integrated protocol, i.e. the required bandwidth, is improved as well. This is due to the fact that

only the short-term public key m and one part of the signature t need to be transmitted as opposed to the key and both parts of the signature (r, t) as is necessary in the combined DH-DSS protocol. In Table 2, we compare the bandwidths of quadratic LFSR-based, cubic LFSR-based, finite field and elliptic curve implementations of the combined and integrated DH-DSS protocols. We list general results as well as results for implementations that achieve a security level comparable to a 1024 bit P and 170 bit q DSA or a 170 q ECDSA implementation. We refer to such a security level as 1024 bit security for short in the following. Note that q is a 170 bit long number in all implementations. In Table 2, however, we do not state results for an implementation of the integrated protocol for the DH-DSA and ECDH-ECDSA cases since these protocols have been shown to be insecure.

We can observe from Table 2 that by implementing the integrated LUC, GH, and XTR DH-DSS protocols instead of their analogue LFSR-based combined protocols, we can achieve a bandwidth reduction of 30 % for GH, 20 % for LUC and 12 % for XTR, respectively. With respect to the total bandwidth requirements, LUC and XTR have better performance than GH. However, the special feature of GH is that both parties share two sequence terms K_1 and K_2 (Equation (CSK-2)) after the protocol execution, as opposed to one in LUC and XTR cases (Equation (QSK-1) and (CSK-1), respectively). As a consequence the ratios $R_1 =$ bandwidth/key size K and $R_2 =$ # po/key size K of a GH implementation are smaller than in LUC and XTR implementations. The former ratio indicates a larger number of session key bits in the GH case which can be very handy in some applications. For instance, in applications where the encryption key needs to be changed more than once every session. In this case, a longer key would avoid the repeated execution of the authentication and key establishment protocol.

If we compare the combined DH-DSA protocol with the LFSR-based integrated protocols in Table 2, we can observe that LUC, GH, and XTR are about 50 %, 37 %, and 62 %, respectively, more efficient in terms of required bandwidth. Comparing now the LFSR-based integrated protocols with the combined ECDH-ECDSA protocol, we can observe from Table 2 that XTR requires the same, LUC 33 % more and GH 66 % more bandwidth. We would like to point out, however, that the two ratios R_1 and R_2, respectively, of quadratic and cubic LFSR DH-DSS implementations are all smaller than in an elliptic curve implementation. The integrated LUC, GH, and XTR protocol, respectively, establish 3, 4, and 2 times, respectively, more session key bits than the combined ECDH-ECDSA protocol.

5.2 Experimental Results and Observations

As shown in the previous section, the integrated protocol requires one primitive operation less for each party than the combined protocol and should be thus about 20 % more efficient in computational time. To fortify this observation, we have implemented the combined and the integrated DH-DSS protocol, where we chose the GH-DSA and GH-DH as LFSR-based signature and key agreement scheme, respectively. We used a slightly modified algorithm than the one in [7, 8] to compute the mixed terms to speed up the computations. We ran all our tests with a fixed set of system parameters that achieve 1024 bit security. The parameters are listed in Table 3. All our results are from an implementation on a Pentium 4, 2.6 GHz, 512 MB RAM desktop computer with the protocols coded in Visual C++ 6.0 using the NTL library [12]. All performance results are taken from 100 runs. We used the pseudorandom number generator of the NTL library to generate long-term and ephemeral secret keys in both protocols and additionally to generate signatures in the combined protocols. The random numbers in our implementation are chosen according to one of the two following methods:

	Integrated DH-DSS	Combined DH-DSS
Bandwidth/party	(m, t)	(m, r, t)
LUC DH-DSA*	$\log Q + \log q$ 680 bit	$\log Q + 2\log q$ 850 bit
GH DH-DSA†	$2\log Q + \log q$ 850 bit	$2\log Q + 2\log q$ 1020 bit
XTR DH-DSA‡	$\log Q + \log q$ 510 bit	$\log Q + 2\log q$ 680 bit
DH-DSA	n/a	$\log P + 2\log q$ 1364 bit
ECDH-ECDSA	n/a	$3\log q$ 510 bit

Table 2: Comparison of bandwidths of some integrated and combined DH-DSS protocols in general and for a 1024 bit security implementation.

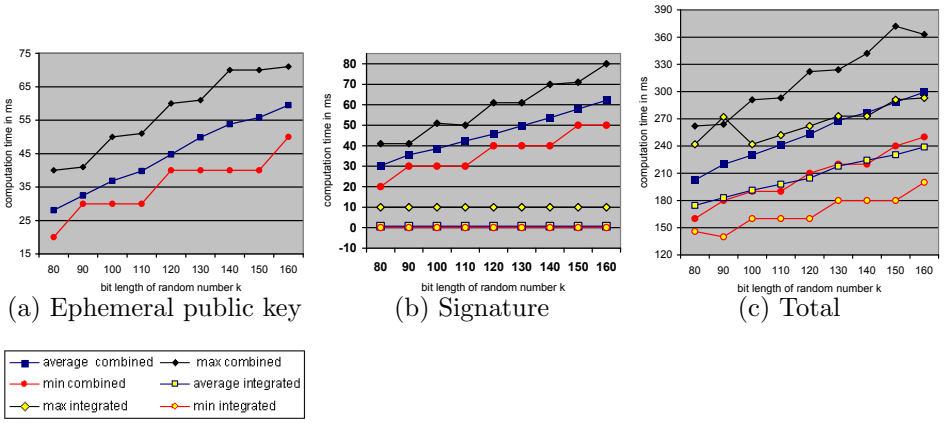
*In LUC $\log Q = \log P/2 \approx 510$ to achieve 1024-bit security.
†In GH $\log Q = \log P/3 \approx 340$ to achieve 1024-bit security
‡In XTR $\log Q = \log P/6 \approx 170$ to achieve 1024-bit security.

$p =$	252410014280206509131998647534662043944278252812238164081281638436419589262881844002472940759520929
$a =$	100967846246663453437323616599547897779132286415320714933049076209148279733077179938397109115148708951
$b =$	206216022644184759815024549954227848108708723659854548174088293500293906237068954063739219293883616268
$Q =$	164705219395020291376758884936962412458513495611

Table 3: Parameters used in GH-DH-DSA implementations to achieve 1024 bit security; with 341 bit prime p, a and b of the characteristic sequence, and period Q.

Method (Rand-i): The number k is randomly chosen out of an interval $[0, Q]$.
Method (Rand-ii): The number k is randomly chosen such that the length l of the random key k is fixed, with $l \in [80, 100, \ldots, 160]$ bits.

We now discuss our experimental results, as documented in Figure 1 and Table 4, in the following. The generation of long-term and ephemeral public keys in integrated and combined protocols as well as the signature computation in the combined protocol require the selection of a random number. Thus, the computation time of these functions depends on the size of the random numbers which is illustrated in Figure 1 (a) for the public key generation and in Figure 1 (b) for the signature computation, respectively, where the random numbers are chosen using method (Rand-ii). We give only one graph for the ephemeral public key generation for both protocols in Figure 1 (a) because the public key generation is identical in both protocols. In this figure, we can observe that the computation time grows linearly with the bit size of the random number. Our performance tests have shown that the computation time of the long-term public keys and the ephemeral public keys are very similar although long-term keys consist of three sequence terms whereas ephemeral keys



(a) Ephemeral public key (b) Signature (c) Total

Figure 1: Computation times (in ms) for (a) ephemeral public key generation, (b) signature generation and (c) for an entire protocol run for the combined and integrated GH DH-DSA protocols.

of just two. Hence, Figure 1 (a) can also be used as good approximation for long-term key computations. Note that we do not consider the generation of long-term keys as part of the protocol, because these computations are only performed as a part of the system set-up.

In Table 4 we summarize the computation times for each protocol step as well as the total computation time of an entire protocol run, where random numbers are chosen according to (Rand-i). We can observe from the table that only the timings for the signature generation differ noticeably for the combined and the integrated protocol. From the table we derive that the integrated protocol is on average $\Delta_{tot,av} = (289.8 - 232.52)\,\text{ms} = 57.28\,\text{ms}$ faster than the combined one. This difference gives a performance gain of approximately 20% and is a result of the different timings for the signature generation in both protocols. Note that the time for computing an ephemeral public key consists of a sequence term calculation only and can thus be considered as the computation time of a primitive operation in a GH scheme. From the table, we get $\text{po} = 57.28\,\text{ms}$ for a randomly chosen k, which is identical to our savings. We can see that the total time for the integrated protocol $t_{tot} = 232.52\,\text{ms}$ corresponds to 4 primitive operations with $4\,\text{po} = 229.12\,\text{ms}$. Similarly for the combined protocol, where $1\,\text{po} = 57\,\text{ms}$ and the total time $t_{tot} = 289.8\,\text{ms}$ corresponds to $5\,\text{po} = 285\,\text{ms}$.

Figure 1 (b) shows the computation time of the signature generation of the combined and integrated GH DSA-DH protocol. We can observe that the required time for signing grows linearly with the bit size $|k|$ of the random signing key in the combined protocol. On the other hand, the computation for signing in the integrated protocol is constant. This can be explained by the re-use of ephemeral public keys as signature parts which makes heavy computations redundant. Thus, signing is virtually for free in the integrated protocol with a maximum computation time of 10 ms. The maximum difference in performance between the two protocols occurs for the maximum key size $|k| = 160$, with $\Delta_{sig,av}(|k| = 160) = (62.18 - 0.7)\,\text{ms} = 61.48\,\text{ms}$.

Figure 1 (c) shows the total computation times of both protocols. We can observe that the computation time grows linearly with the size of the random numbers. Note that the same bit size is used for both random keys, i.e. the ephemeral secret key and the signing key. The maximum gap between both protocols can be observed for maximum random key size with $\Delta_{tot,av}(|k| = 160) = (299.43 - 238.95)\,\text{ms} = 60.48\,\text{ms}$, which equals approximately the maximum time difference for the

		Integrated GH DH-DSA	Combined GH DH-DSA
Computing ephemeral public key	average	57.28	57
	max.	80	70
	min.	40	40
Signing	average	0.7	58.14
	max.	10	71
	min.	0	50
Verifying	average	117.85	118.46
	max.	141	161
	min.	100	100
Computing shared key K	average	56.69	56.2
	max.	91	80
	min.	40	40
Total time/ party	average	232.52	289.8
	max.	322	382
	min.	180	230

Table 4: Computation times (in ms) for individual protocol steps and total protocol execution for combined and integrated GH DH-DSA protocol. The random numbers are chosen out of an interval $[0, Q]$ (Rand-i).

signature generation, i.e. $\Delta_{sig,av}(|k| = 160)$. This again corresponds to a performance gain from about 20% as expected in our theoretical analysis.

In summary, we can conclude that our experimental results confirm the results of our theoretical analysis. The experimental results show that signing is virtually for free and the integrated GH DH-DSS protocol is indeed 20% faster in computational time than the combined GH DH-DSS protocol. Furthermore, the results showed that our assumption that we only need to consider primitive operations to analyze the computational performance of protocols was correct and the required time to perform all other computations is negligible.

Remark. We also implemented the combined DH-DSS protocols for DH-DSA and ECDH-ECDSA using the same platform, where we used the Crypto++ library [4] for the elliptic curve implementation. The experimental results of these two implementations showed that considering only primitive operations is sufficient for a performance analysis and the computation times of all other operations are negligible. In our implementation a primitive operation in DH-DSA, i.e. the execution of one modular exponentiation, takes in average po(DH-DSA) = 20.23 ms, where 1 po in ECDSA-ECDH, i.e. one scalar multiplication, takes in average po(EC-ECDSA) = 16.33 ms for the random numbers chosen according to (Rand-ii). Consequently, both computations are faster than a primitive operation in our GH implementation. However, the GH code that we used is not optimized at all. The goal of the paper is to show the significant performance gain that can be achieved by implementing an integrated DH-DSS protocol. Since LFSR-based schemes use smaller subfields for computations their performance can be expected to be better than of finite field implementations. The same optimization methods that were used to improve the performance of DSA and ECDSA in the past decades can be applied to LFSR-based schemes as well. This kind of optimization is out of the scope of this paper and remains the subject future research.

6 Efficient Protocols from Integrating DH into DSS in Existing Protocols

The presented integrated LFSR-based DH-DSS protocols provide authenticated key agreement between two parties while resisting known-key attacks. However, if we would like an efficient authenticated key exchange (AKE) protocol that provides more security properties or a full security proof we can apply the same method of integrating DH key exchange into DSS to any DH-DSS-based AKE protocol that has the desired properties. In that way we obtain a more efficient protocol while maintaining all the security properties of the original protocol. The performance gain is significant as previously shown and the method can be used to develop efficient AKE protocols that are well-suited for wireless networks and constrained devices. If mutual authentication is also desired, the same method can be applied to secure authentication and authenticated key exchange (MAAKE) protocols, such as the IKE protocol [9] for instance. To transform a secure AKE or MAAKE protocol into a more efficient protocol we only need to perform two steps:

1. Replace the DH key exchange and DSS of the original protocol by an LFSR-based DH scheme and DSS, respectively, such as LUC, GH, or XTR.
2. Integrate the DH key exchange into the DSS as described in our presented LFSR-based integrated protocols.

6.1 The Integrated Signature-Based IKE Protocol

We now demonstrate the general approach of applying the integrated DH-DSS protocol to other AKE and MAAKE protocols on the signature-based IKE protocol to obtain an integrated version of the protocol. To do so, we use a quadratic or cubic LFSR-based DH key exchange and DSS in the original protocol. Then, we integrate the DH key exchange into the DSS. Since IKE was proven to be secure in [3], the resulting integrated IKE protocol remains secure while showing better computational and communications performance. The transformation is done in a straightforward way and we illustrate it for the basic SIGMA protocol, which is a simplified version of IKE's signature-mode. Note that this result can be easily adopted to the actual signature-based IKE protocols. The protocol messages of an LFSR-based integrated basic SIGMA protocol are summarized in Protocol 3 and we discuss the protocol steps in detail in the following. As usual, we assume that both parties A and B have authentically exchanged their long-term public keys y_A and y_B prior to the protocol execution.

Protocol 3. Integrated Basic SIGMA Protocol

System Parameters: Message authentication code family MAC and pseudo random function PRF.

Protocol Messages:

1. $A \longrightarrow B : s, f_{k_A}$
2. $A \longleftarrow B : s, f_{k_B}, ID_A, t_B, MAC_{k_1}(\text{"1"}, s, ID_A)$
3. $A \longrightarrow B : s, ID_B, t_A, MAC_{k_1}(\text{"0"}, s, ID_A)$

Shared Key: MAC key $k_1 = PRF_K(1)$ and session key $k_0 = PRF_K(0)$, with K computed according to QSK-1 for LUC, CSK-2 for GH or CSK-1 for XTR.

For brevity, we consider the quadratic and cubic LFSR-based implementations simultaneously, where we write the information for cubic cases in brackets. The system parameters and long-term keys are the same as given in Section 3.1 (3.2). Party A performs steps QS-1 and QS-2 (CS-1 and CS-2) to choose its ephemeral secret key k_A and compute the ephemeral public key f_{k_A}. A then sends a session ID s and f_{k_A} to user B in the first protocol flow. After checking that the session ID is fresh, the receiver, party B, performs steps QS-1 and QS-2 (CS-1 and CS-2) to compute its ephemeral keys s_{k_B} and f_{k_B}. B then forms $h_B = h(\text{"1"}, s, f_{k_A}, f_{k_B})$, where $h(\cdot)$ is a hash function, and signs h_B by performing step QS-3 (CS-3). Now, B computes the shared key K according to QSK-1 (CSK-2 or CSK-1 for GH or XTR, respectively), derives the MAC key $k_1 = PRF_K(1)$ and the session key $k_0 = PRF_K(0)$ using K and the pseudorandom function PRF, and computes the MAC over the concatenated string formed from a string "1", s and A's public identifier ID_A. B then returns the session ID, f_{k_B}, A's ID, one part of the signature t_B, and the MAC value in the second message flow. Upon receiving the data from the second flow, A forms h_B using its own and the received ephemeral public key, the session ID and a string "1", sets $r_B = s_{k_B}$, and verifies the signature by performing step QV-1 to QV-4 (CV-1 to CV-4). Now A computes K according to QSK-1 (CSK-2 or CSK-1 for GH or XTR, respectively), derives k_1 and k_0, and verifies the received MAC value. If both verifications were successful A forms $m_A = h(\text{"0"}, s, f_{k_B}, f_{k_A})$ and signs it according to step QS-3 (CS-3). Then A computes the MAC over the concatenated string formed from a string "0", the session ID and its own ID and sends s, ID_B, t_A and the MAC value to B in the third protocol flow. Upon receiving the last message, B forms h_A and r_A and verifies the signature according to steps QV-1 to QV-4 (CV-1 to CV-4). B also verifies the MAC. If both verifications were successful, B accepts. Both parties share the session key k_0 after completing Protocol 3. Note that in the GH case both parties derive a bipartite session key $K = (K_1, K_2)$, where K_1 and K_2 might be directly used as k_0 and k_1, respectively.

We can observe that the integrated basic SIGMA protocol is more efficient than the original protocol in terms of computational costs, since we can save one primitive operation for computing the signature part r, and in terms of bandwidth, because we only need to send one signature part t as opposed to both (r, t), as in the original protocol.

7 Conclusions

The results of this paper are three-fold. First, we have shown that we can securely integrate a DH-like key agreement into a DSS, as proposed by Arazi, if we use second-order or third-order characteristic sequences, such as LUC, GH, and XTR, instead of crypto systems employing exponential functions or elliptic curves. The advantage of the proposed integrated LFSR-based DH-DSS protocols are savings in computational costs and bandwidth as well as resistance to the known-key attack proposed by Nyberg and Rueppel.

Second, we have demonstrated the significant advantage in performance of our integrated protocol over a combined protocol, by a theoretical analysis and experimental results. Our integrated protocol is 20% faster in terms of computational cost and up to 30% more efficient in with respect to the bandwidth as the conventionally combined DH-DSS protocol.

Last, we have shown how our integrated DH-DSS protocol can be used to transform other DH-DSS-based AKE and MAAKE protocols to increase their computational performance and reduce their bandwidth while maintaining all security features of the original protocols. Thus, we developed a tool to create secure efficient protocols from existing protocols so that the obtained protocols are

well-suited for wireless mobile networks. We demonstrated the approach on the IKE protocol and presented an integrated IKE protocol that is more efficient than the original IKE protocol while maintaining all its security properties.

As previously mentioned, LFSR-based schemes are using smaller subfields for computation than schemes that use the exponential function, such as the DSA. In addition, the schemes can be efficiently implemented in hardware. These facts combined with the results of this paper make the presented LFSR-based protocols attractive for an implementation in wireless and all other constrained environments.

References

[1] B. Arazi. Integrating a Key Distribution Procedure into the Digital Signature Standard, *Electronics Letters*, 1993, vol. 29, no. 11, pp. 966-967.

[2] D. Brown and A. Menezes. A Small Subgroup Attack on Arazi's Key Agreement Protocol, *Technical Report, Centre for Applied Crypographic Research, CORR 2001-50*, University of Waterloo, 2001.

[3] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-Based Key-Exchange Protocol, *Advances in Cryptology - CRYPTO '2002*, M. Yung (Ed.), LNCS 242, Springer-Verlag, 2002, pp. 143-161.

[4] Crypto++ ®Library. Version 5.2.1. Free download available at `http://www.eskimo.com/~weidai/cryptlib.html`.

[5] S.W. Golomb. Shift Register Sequences, Revised Edition, Aegean Park Press, May, 1982.

[6] G. Gong and L. Harn. Efficient Lucas-type Public Key Cryptosystems, *Proceedings of 1996 International Conference on Cryptology and Information Security*, 1996.

[7] G. Gong and L. Harn. Public-key Cryptosystems Based on Cubic Finite Field Extensions, *IEEE Trans. on Inform. Theory*, 1999, vol. 45, no. 7, pp. 2601-2605.

[8] G. Gong, L. Harn, and H. Wu. The GH Public-Key Cryptosystem, *Proceedings of Selected Areas in Cryptography (SAC) 2001*, LNCS 2259, Springer-Verlag, 2001, pp. 284-300.

[9] IETF- The Internet Engineering Taskforce, RFC 2409 on IKE, "The Internet Key Exchange (IKE)", Proposed Standard, 1998.

[10] L. Harn and M. Mehta. Integrating Diffie-Hellman Key Exchange into the Digital Signature Algortihm (DSA), *IEEE Communications Letters*, 2004, vol. 8, no. 3, pp. 198-200.

[11] A.K. Lenstra and E.R. Verheul. The XTR Public Key System, *Advances in Cryptology - CRYPTO '2000*, LNCS 1880, Springer-Verlag, 2000, pp. 1-9.

[12] NTL: A Library for doing Number Theory. Version NTL 5.3.2. Free download available at `http://www.shoup.net/ntl/`.

[13] K. Nyberg and R.A. Rueppel. Weaknesses in Some Recent Key Agreement Protocols, *Electronics Letters*, 1994, vol. 30, no. 1, pp. 26-27.

[14] J.L. Massey. Shift-Register Synthesis and BCH Decoding, *IEEE Trans. on Inform. Theory*, 1969, vol. 15, no. 1, pp. 81-92.

[15] W.B. Mueller and W. Noebauer. Cryptanalysis of the Dickson-Scheme, *Advances in Cryptology - EUROCRYPT '85*, LNCS 219, Springer-Verlag, 1986, pp. 50-61.

[16] P. Smith. A Public-Key Cryptosystem and a Digital Signature System Based on the Lucas Function Analogue to Discrete Logarithms, *Advances in Cryptology - ASIACRYPT '94*, LNCS 917, Springer-Verlag, 1995, pp. 357-364.

[17] M. Stam and A.K. Lenstra. Speeding Up XTR, *Advances in Cryptology - ASIACRYPT '01*, LNCS 2248, Springer-Verlag, 2001, pp. 125-143.