A Low-Cost Solution for Frequent Symmetric Key Exchange in Ad-hoc Networks

Markus Volkmer and Sebastian Wallner

Technische Universität Hamburg-Harburg – Technische Informatik VI
Schwarzenbergstraße 95, D-21073 Hamburg, Germany
{markus.volkmer,wallner}@tuhh.de

Abstract: Next to authentication, secure key exchange is considered the most critical and complex issue regarding ad-hoc network security. We present a low-cost, (i.e. low hardware-complexity) solution for feasible frequent symmetric key exchange in ad-hoc networks, based on a Tree Parity Machine Rekeying Architecture. A key exchange can be performed within a few milliseconds, given practical wireless communication channels and their limited bandwidths. A flexible rekeying functionality enables the full exploitation of the achievable key exchange rates. Characteristics of a standard-cell ASIC design realisation as IP-core in 0.18μ-technology are evaluated.

1 Key Exchange in Ad-hoc Networks

Authentication and key exchange are of supreme importance with regard to security and secure routing in ad-hoc networks. Such networks typically lack infrastructure, which penalises approaches needing a central authority like a trust centre or another third trusted party as e.g. in ID-based cryptosystems. The dynamic and ephemeral network topology demands frequent key exchanges (cf. [KA02] for an unbiased survey). In this regard, cryptographic methods with appropriate computational efficiency, that also consider a certain message or protocol overhead, are inevitable. In ubiquitous and pervasive computing applications, such as sensor networks, RFID-systems or Near Field Communication (NFC), the devices in use as nodes of the network often impose severe size limitations and power consumption constraints. Consequently, the available size for additional cryptographic hardware components is limited as well [KA02, St03, WGP03].

Public key or threshold cryptography are in general computationally intensive and a distributed certificate authority does not address the resource limitations of devices in ad-hoc networks. Asymmetric algorithms for key exchange like RSA and El Gamal perform computationally intensive arithmetics, typically implemented in software on limited microcontrollers. The state-of-the-art is represented by (Hyper-)Elliptic Curve Cryptography (see e.g. [PWP03]). Retaining the security, these representations decrease the size of numbers in the necessary arithmetic operations, but increase the operations' complexity. Still, a (frequent) key exchange in practice often remains of prohibitive cost.

2 Tree Parity Machine Rekeying Architecture

The Tree Parity Machine Rekeying Architecture (TPMRA) is a small hardware solution for frequent symmetric key exchange in ad-hoc networks, based on a 'hardware-friendly' algorithm for secure key exchange by synchronisation of Tree Parity Machines [KKK02]. It is functionally separated into the TPM Unit and its control state machine, the Key Handshake and Bit Package Control and a Watchdog timer (Figure 1).

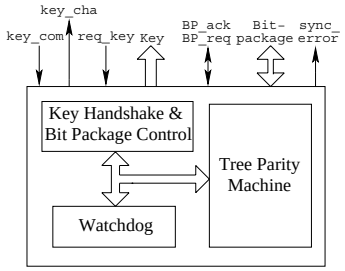


Figure 1: Basic diagram of the Tree Parity Machine Rekeying Architecture.

A mutual adaptation process between the two parties A and B realises the exchange protocol without involving large numbers and methods based on number theory. In the following, the notation A/B denotes equivalent operations for the parties A and B, while a single A or B denotes an operation specific to one of the parties. The TPM Unit generally comprises K hidden units ($1 \leq k \leq K$) in a single hidden-layer with non-overlapping inputs. Each hidden unit receives different N inputs ($1 \leq j \leq N$), leading to an input of size $K \cdot N$. The components are pseudo-random variables realised by equally initialised Linear Feedback Shift Registers (LFSR). A single output $O^{A/B}(t) \in \{-1, 1\}$, given bounded weights $w_{kj}^{A/B}(t) \in [-L, L] \subseteq \mathbb{Z}$ (from input unit j to hidden unit k) and common random inputs $x_{kj}(t) \in \{-1, 1\}$, is calculated as

$$O^{A/B}(t) = \prod_{k=1}^{K} y_k^{A/B}(t) = \prod_{k=1}^{K} \sigma\left(\sum_{j=1}^{N} w_{kj}^{A/B}(t)\, x_{kj}(t)\right) . \tag{1}$$

The common random inputs can also be kept secret between the parties, yielding authentication. σ is a party-specific modified sign-function, that defines an agreement between the two parties on an opposite sign in case of a sum of zero.

Initial weights can either be device-specific or they can be provided by an additional application-specific device, e.g. a thermal noise device. The outer product in eq. 1 can be realised without multiplication. The product within the sum only changes the sign of the weight. Thus, an adder is the most complex structure to be implemented. The test for the sign or the test on equality to zero are easily done in hardware.

We implemented the *bit package* generalisation of the protocol and the parallel-weights version (cf. [KKK02]). Parties A and B start with an individual randomly generated secret initial weight vector $w_{kj}^{A/B}(t_0)$. After a set of $b > 1$ presented inputs, where b

denotes the size of the bit package, the corresponding b TPM outputs (bits) $O^{A/B}(t)$ are exchanged over the public channel in one package. This reduces output communication and is advantageous for practical communication channels with protocol overhead. The Key Handshake and Bit Package Control handles the key transmission with an encryption unit and the bit package exchange process with the other party. It is accomplished by partitioning the parity bits from the TPM unit in tighter bit slices. We chose a bit package length of 32 bit (`Bit Package`) and the bit package exchange process uses a simple request/acknowledge handshake protocol (`BP_ack`, `BP_req`). The amount of registers needed for storage increases in the bit package variant, finally imposing a tradeoff area vs. speed. The b sequences of hidden states $y_k^{A/B}(t) \in \{-1, 1\}$ are stored for the subsequent adaption process.

Weights are adapted, using the b outputs and b sequences of hidden states:

$$w_{kj}^{A/B}(t) := \begin{cases} w_{kj}^{A/B}(t-1) + O^{A/B}(t)\; x_{kj}(t) & , O^A(t) = O^B(t) \wedge \\ & \quad O^{A/B}(t)\; y_k^{A/B}(t) > 0 \\ w_{kj}^{A/B}(t-1) & , \text{otherwise.} \end{cases} \tag{2}$$

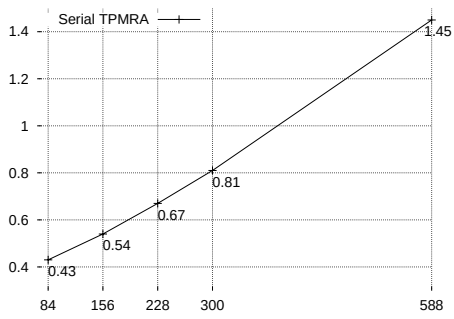
The sign-operations and additions are well suited for a hardware implementation. Updated weights with $|w_{kj}^{A/B}(t)| > L$ are bound to stay in the maximum range $[-L, L] \subseteq \mathbb{Z}$ by reflection onto the boundary values.

Synchrony is present when both parties adapted to produce each others outputs and is achieved only for *common* inputs. Thus, keeping the common inputs secret between A and B can be used to have an authenticated key exchange. There are $2^{KN} - 1$ possible inputs in each iteration, yielding as many possible initialisations for a pseudo random number generator. Synchronisation time is distributed, is proportional to L^2 and peaked around 400 for the parameters given in [KKK02]. The synchronisation criterion basically comprises a counter, to test on successive equal outputs in a sufficiently large number of iterations. This excludes equal outputs occurring by chance. The Watchdog timer supervises the number of interactions needed for a key-exchange between two parties. If there is no synchronisation within a specific time, a signal (`sync_error`) indicates a synchronisation error. Once synchronous, parties remain synchronised (see eq. 2) and common weight-vectors are present in both TPMs. These weights have not been communicated and can be used as a common key.

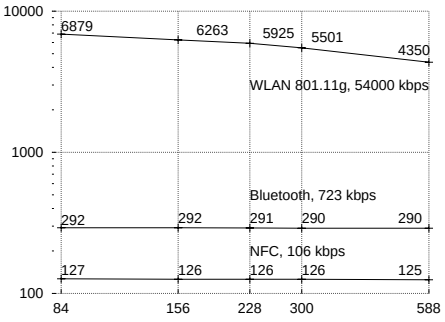
We minimise the key lifetime as much as possible employing *immediate rekeying*, i.e. a frequent key exchange. Such a rekeying process is often of prohibitive cost due to inefficiencies in classical key exchange methods. Yet, using the TPM principle allows for efficient rekeying (see also section 3). In our hardware design, an external unit can demand a key exchange service. The TPMRA can continuously synchronise new keys, as long as data needs to be exchanged securely. Once an en-/decryption unit uses the first key, synchronisation can be triggered again to always provide a new key. Due to different computation cycles between two key exchange parties, the rekeying procedure employs a handshake protocol with a key request (`req_key`), a key changed (`key_cha`) and a key commit (`key_com`) (see Figure 1). The internal bus (`Key`) hands over the key to an encryption unit when the synchronisation process is finished. Keys can so be exchanged at a maximum rate subject to hardware constraints and available channel bandwidth.

3 Characteristics of an ASIC Implementation

We designed and simulated a parameterisable serial TPMRA using VHDL. The underlying process was a 0.18μ six-layer CMOS process with $1.8V$ supply voltage based on the UMC library. The synaptic summation is performed by Time Devision Multiple Access (TDMA) of an L-bit adder. A standard cell ASIC prototype realisation was build to verify the suitability of the TPMRA as IP-Core in devices for ad-hoc networks. The linear complexity of the protocol scales with the size $K \cdot N$ of the TPM structure, which defines the size $K \cdot N \cdot L$ of the key. We chose $K = 3$, a maximal $N = 49$ and $L = 4$, leading to a key size of 588 bit. The chosen N allows a significant key length and still keeps the average synchronisation time low (cf. [KKK02]).



(a) Chip-area $[mm^2]$ vs. key length $[bit]$.



(b) Worst case average key exchange rate $[Hz]$ vs. key length $[bit]$ for three channels/protocols.

Figure 2: Post-synthesis results: (a) Chip-area (logic) vs. key length. (b) shows worst case average key exchange rate (log-scaled) vs. key length (worst case of an avg. synchronisation time of 400 iterations for a 588 bit key assumed for all key lengths) and a selection of typical protocols with their bandwidths: Bluetooth (190 bit minimum packet length), WLAN 801.11g (512 bit minimum packet length) and a Near Field Communication channel (64 bit minimum packet length). All data refers to a UMC 0.18 micron six-layer standard cell process realisation.

The area (Figure 2a) of the TPMRA realisation scales approximately linear due to the linear complexity of the adders and ranges around one square-millimetre for the investigated key sizes. Note, that most of the area is consumed by the bit packaging, because of the necessary storage of the inputs for the adaption (see section 2).

The nearly linear decrease of the throughput (Figure 2b) for all channels is due to the linear complexity of the key exchange protocol. We assumed the maximally achievable clock frequency with regard to each key length, which can be achieved by Digital Phase Lock Loop (DPLL), regardless of the systems clock frequency. Furthermore, we appoint an average synchronisation time of 400 iterations for all key lengths, although it is always less than the size of the key (a worst-case scenario). For every protocol, we used

the minimum available packet length due to our bit packages of 32 bit. A comparison among the different communication channels indicates different slopes of the calculated (approximately linear) throughput characteristics. They denote the rising influence of the bit packaging calculations at smaller key lengths for channels of higher bandwidth such as WLAN. Thus, the slope of the WLAN throughput characteristic is significantly higher than for Bluetooth and NFC. As expected, the influence of the channel bandwidth significantly determines the performance of the key exchange protocol.

4 Summary and Outlook

We presented a low-cost solution for frequent symmetric key exchange for devices with severely limited resources, as they are typical in ad-hoc networks. The proposed Tree Parity Machine Rekeying Architecture's silicon area lies within a square-millimetre and allows to exchange keys of practical size within milliseconds. The relatively small size of the TPMRA allows an implementation in general embedded systems with only small overhead. Thus we promote it as an IP-core for wireless ad-hoc network devices. A detailed discussion of the security of the method itself is beyond the scope of this paper. All currently known attacks, however, require significant hardware expenses, are successful only with a certain probability and all refer to a non-authenticated key exchange. A future fully serial realisation of the architecture will use TDMA of a single hidden unit. This further decreases the silicon area consumption but at the cost of an increase in necessary cycles for one output bit. Key exchange between multiple parties, to secure group communication in a broadcast or multicast scenario, is also possible and under investigation. An efficient authentication and key management is needed here, due to frequent key exchanges on join or leave actions. The integration into concrete devices of an ad-hoc network and its practical evaluation is also subject to future work.

References

[KA02] Khalili, A. und Arbaugh, W. A.: Security of wireless ad-hoc networks. *ACM Computing Surveys*. 2002. (submitted, http://www.cs.umd.edu/~aram/wireless/survey.pdf).

[KKK02] Kanter, I., Kinzel, W., und Kanter, E.: Secure exchange of information by synchronization of neural networks. *Europhysics Letters*. 57(1):141–147. 2002.

[PWP03] Pelzl, J., Wollinger, T., und Paar, C.: Low cost security: Explicit formulae for genus-4 hyperelliptic curves. In: *10th Annual Workshop on Selected Areas in Cryptography (SAC 2003)*. Springer Verlag. 2003.

[St03] Stajano, F.: Security in pervasive computing. In: *Proc. of the 1st International Conference on Security in Pervasive Computing (SPC 2003)*. volume 2802 of *LNCS*. S. 1. Springer Verlag. 2003.

[WGP03] Wollinger, T., Guajardo, J., und Paar, C.: Cryptography in embedded systems: An overview. In: *Proc. of the Embedded World 2003 Exhibition and Conference*. S. 735–744. Nürnberg, Germany. Feb. 18-20 2003. Design & Elektronik, Nürnberg.