Social Key Exchange Network – From Ad-Hoc Key Exchanges to a Dense Key Network

Dirk Achenbach David Förster Christian Henrich Daniel Kraschewski
Jörn Müller-Quade

Institut für Kryptographie und Sicherheit (IKS), Fakultät für Informatik,
Karlsruher Institut für Technologie (KIT)

{achenbach,henrich,kraschewski,mueller-quade}@kit.edu
david@dfoerster.de

Abstract: Security of public key cryptography is steadily threatened by advancements in algorithmics and computing power. In this work we propose a novel approach to long-term secure key exchange based on security assumptions that are independent of strong complexity assumptions.

We present a key propagation scheme that sets up a network of distributed keys. Whenever two parties meet, they exchange new keys (e.g. using near field communication) and pass on all keys received so far. This establishes a dense key network growing and spreading with each meeting of protocol participants. Even two parties that have never met in person can use this network to obtain a common secret. A notable security feature of our scheme is the anonymity of the established keys, making it hard for an adversary to track movements of protocol participants.

Keywords: Key Propagation, Mobile Ad-Hoc Networks, Key Exchange.

1 Introduction

The desire for encrypted communication, secure against any eavesdropper, is as old as the concept of mounted messengers. But, in order to establish an encrypted communication channel, both communication partners must initially share a key that is unknown to the adversary. Given a trusted party (the key server) in the center of a star network where every party has a key in common with the key server, key exchange can be done by the Kerberos or Needham-Schroeder protocol [SNS88, NS78], for instance. However, the key server is a single point of failure and compromising it destroys any secrecy of session keys.

With the discovery of public-key cryptography, key exchange procedures over an insecure channel were introduced [Mer78, DH76]. Nonetheless, these approaches build on strong complexity assumptions and therefore are steadily threatened by advances in algorithmics and computing power. For example, in 1977 Rivest estimated that factoring a 169-digit RSA modulus would take about 40 quadrillion years [Gar77]; in 1994 it was done in eight months, employing a quadratic sieve algorithm and using the idle cycles of computers worldwide [AGLL94]. Additionally, a theoretical threat emanates from quantum algorithms for integer factoring and computation of discrete logarithms [Sho94].

ISBN 978-3-88579-286-4

1.1 Our Contribution

In this work we propose a novel approach to key exchange. The basic idea of our *Social Key Exchange Network* (SOKEN) is to establish a network of shared secrets exchanged and distributed independently of communication desires. Whenever participants meet, they exchange key material and forward all key material they have received so far—we propose to utilize mobile communication devices equipped with short-range radio systems (e.g. Blutooth-enabled cellphones). Using the key material from the network two participants can establish a common secret over a public channel, even if they have never met before. This is possible through the forwarding of key material and the assumption that the adversary has not observed all of the meetings when the key material was exchanged.

SOKEN is a distributed system and thus robust against single corruptions of participants. It also provides long-term security: A secret session key derived from the network will remain secret no matter how many participants other than the two communicating parties are corrupted afterwards. As our security assumptions are skew to the strong complexity assumptions of conventional public-key schemes, we propose to combine our approach with such a key exchange, thus achieving a maximum of security.

1.2 Related Work

Location-limited channels have been exploited for authentication in wireless sensor networks [ACP04] as well as ad-hoc wireless networks [BSSW02].

The concept of using a vast number of key servers to achieve long-term security was explored in the virtual satellites approach [Rab03]. There, it is also assumed that the adversary has limited access to the many web sites providing key material.

In general, mobile ad-hoc networks (MANETs) have become a topic of interest in telematics and related fields. Su et al. [SCP+04] and Chaintreau et al. [CHC+05] have explored Pocket Switching Networks, a special type of MANET that employs pocket-sized devices. They examined how human mobility can be exploited to route messages between nodes that are not connected directly in such networks.

Geambasu et al. introduced Vanish [GKLL09], a technique for cryptographically making data inaccessible after a certain amount of time. They also follow a distributed approach: A cryptographic key is split and dispersed on a peer-to-peer sharing network where it degrades after some pre-determined time.

2 The Social Key Exchange Network (SOKEN)

In this section we present our novel approach. For ease of presentation we start with a rudimental version (Section 2.1) and then refine it step by step (Section 2.2).

Protocol: MEET

1. The sender generates a new initial key $(ID, k, 0)$ with uniformly random ID and key value, sends it to the receiver and saves it to his keyring $\mathcal{I}$.
2. For each (ID, k, ctr) in his keyring $\mathcal{C}$, the sender also forwards $(ID, \mathrm{h}(k), ctr+1)$.
3. The receiver stores each received key in his keyring $\mathcal{C}$.

Figure 1: The basic version of the protocol MEET. It is always executed twice in parallel, each party posing as sender once.

2.1 Basic Key Exchange Network

We differentiate between two phases: During the *Preparation Phase* key material is generated and distributed among the participants of the network. This stage is unaware of how people may want to communicate in the other phase, the *Communication Phase*. In the latter a common secret is established from the key material exchanged so far. This can be used to strengthen any direct key exchange.

Preparation Phase. This is the regular operation mode of SOKEN. Throughout its lifetime the network serves to generate and spread new key material. Whenever two parties meet, the same process is carried out: Each one generates and sends a new *initial key*, then they both forward keys received so far from other parties. Though, in order to constrain the disclosure of key material, a cryptographic hash function is applied to every key before it is passed on. (We call the resulting key a *derived key* or simply *derivate*.) That way, nodes can compute keys that they have passed on before but not the other way round. This encourages the passing-on of derivates, because it does not disclose the received key.

Users of the network hold two key rings: *initial keys* $\mathcal{I}$ and *communication keys* $\mathcal{C}$. The keyring $\mathcal{I}$ is used to store the keys a party created itself. The keyring $\mathcal{C}$ contains keys that have been received from other participants and will be used to initiate communication with the originator of the respective key. Each key consists of a (preferably) unique ID, the actual key value k and a counter that denotes how often the key has been hashed so far. Formally, we write a key record as a 3-tuple (ID, k, ctr). The protocol MEET (s. Figure 1) implements the exchange and forwarding of keys. Whenever two parties meet, it is executed twice; each party poses as sender once.

Communication Phase. This phase is initiated on demand to establish a common secret between two parties. It is asynchronous and independent of the Preparation Phase (given sufficient key material has been exchanged so far). Even if the two parties have not directly exchanged a key before, no interaction with any third party is required during the Communication Phase.

Two parties can extract a common secret from the network as follows. At first they have to identify all common keys, i.e. keys that originated at one of them and were passed on to the other. Then for each key the party that has the derivate announces the corresponding counter value, thereby enabling the other party to calculate the derivate from its initial key.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

Protocol: CALL

1. The sender announces the IDs of all keys in his keyring $\mathcal{I}$, say $ID_1, \ldots, ID_{|\mathcal{I}|}$.
2. The receiver checks his keyring $\mathcal{C}$ for these IDs and announces the list of all matches and the according counters, say $(ID_{\nu_1}, ctr_{\nu_1}), \ldots, (ID_{\nu_n}, ctr_{\nu_n})$. For key confirmation purposes, he also sends along a hash of each key value, i.e. $\mathrm{h}(k_{\nu_1}), \ldots, \mathrm{h}(k_{\nu_n})$.
3. For each received tuple (ID_i, ctr_i), the sender computes k_i by hashing the corresponding initial key for ctr_i times. Then he checks for correctness using the provided hash value $\mathrm{h}(k_i)$. All correct keys are combined into the session key, e.g. by hashing them together, and the corresponding IDs are sent back to the receiver.
4. The receiver analogously combines the corresponding keys into the session key.

Figure 2: The basic version of the protocol CALL. It is always executed twice in parallel, each party posing as sender once. Note that instead of separately producing two different session keys, as well all correct key values of both protocol runs can be combined into one session key.

All of those derived keys are then combined in a secure manner to form a session key. The protocol CALL (s. Figure 2) implements the search for common key IDs and an additional key confirmation to sort out faulty keys. It is important to note that only cryptographic hashes but no actual key values are exchanged during this protocol.

2.2 Refinements

The basic protocol versions do not yet provide all promised security features. Above all, it is very unsatisfying that once a party P received a key (ID, k, ctr) it will forward the same derivate $(ID, \mathrm{h}(k), ctr + 1)$ to many different parties, so that just by executing the protocol MEET with P an adversary can learn all derivates that ever went through P's hands. The same problem occurs when P's key storage device is stolen. Therefore, we propose a more sophisticated process of key derivation and forwarding. Firstly, instead of always forwarding the same derived key value $\mathrm{h}(k)$, it is preferable to vary the hash function; technically this can be done by padding k. Secondly, whenever a party forwards a derivate of some key, the key should also be replaced by another drivate of itself in that party's device. Of course, one has to store and pass along with each derivate all the information that is needed to reconstruct this derivate from the corresponding initial key.

Another problem is that in the protocol CALL each party publicly announces all IDs of its initial keys. This somehow contradicts untraceability of protocol participants, as an adversary can easily get hold of a party P's initial keys' IDs. Then, by initiating the protocol CALL with arbitrary parties and sending the ID list received from P, the adversary can systematically derive a survey of P's acquaintances. This attack can be hampered by using salted hash values of the key IDs instead of plain IDs for the comparison step. This way, when the salt value is specified by the receiver of the communication request, an adversary can no longer actively set up a systematical survey of an arbitrarily chosen parties' acquaintances. In particular, the initiator of the protocol CALL can no longer pass

Protocol: MEET

1. The sender generates an initial key $(ID, k, [\,])$ with uniformly random ID and key value and empty padding list, sends it to the receiver and saves it to his keyring $\mathcal{I}$.
2. For each key (ID, k, S) in his keyring $\mathcal{C}$, the sender computes and forwards $(ID, \mathrm{h}(k\|1), [S, 1])$ and then replaces it by $(ID, \mathrm{h}(k\|0), [S, 0])$ in his key storage.
3. The receiver stores each received key in his keyring $\mathcal{C}$.

Protocol: CALL

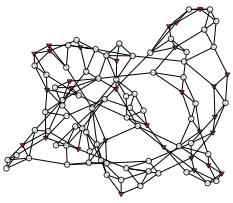
0. Let $nonce$ denote a random nonce that is chosen by the receiver of the communication request.
1. The sender announces salted hash values of the IDs of all keys in his keyring $\mathcal{I}$, namely $\mathrm{h}(ID_1\|nonce), \ldots, \mathrm{h}(ID_{|\mathcal{I}|}\|nonce)$.
2. The receiver computes and announces the index set $\{\nu_1, \ldots, \nu_n\}$ of all these IDs that also occur in his keyring $\mathcal{C}$. He also announces the corresponding padding lists $S_{\nu_1}, \ldots, S_{\nu_n}$ and hashed key values $\mathrm{h}(k_{\nu_1}), \ldots, \mathrm{h}(k_{\nu_n})$.
3. For each received index ν, the sender retrieves the initial key $(ID_\nu, \tilde{k}_\nu, [\,])$ from his key ring $\mathcal{I}$ and, given that $S_\nu = [b_1, \ldots, b_m]$, recomputes the respective derivate $k_\nu = \mathrm{h}(\ldots \mathrm{h}(\tilde{k}_\nu\|b_1) \ldots \|b_m)$. Then he uses the received hash value $\mathrm{h}(k_\nu)$ for a consistency check and announces whether this key is correct.
4. Sender and receiver each locally combine all correct keys into the session key.

Figure 3: The more sophisticated versions of our protocols MEET and CALL. Both protocols are always executed twice in parallel, each party posing as sender once. However, the $nonce$ in the protocol CALL is always chosen by the receiver of the communication request.

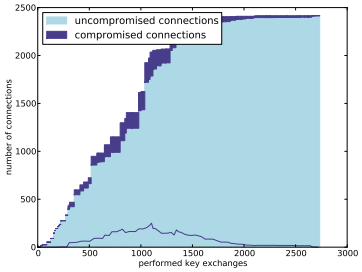
on a previously received ID list as his own. (See Figure 3 for the refined protocols.)

3 Parameters

The probability of a key being compromised increases with the number of parties that were involved in its forwarding, i.e. its hop count. Therefore, it is desirable to introduce a maximum hop count to save memory. Such a maximum hop count does not have to be a global parameter of the network, but each participant can choose a custom value. Further note that the padding lists of derivates can be run-length encoded very efficiently, since the number of non-zero entries is limited by the maximum hop count minus one. We suggest a maximum hop count of 6 (motivated by the small-world phenomenon [Mil67]) and to implement the padding list by five 8-bit-counters, thus limiting the number of forwardings to 256 on each hop distance level. As our main cryptographic primitives are hash functions, the length of a key depends on the used hash function. A reasonable choice would be SHA-1, for example, whose output length is 160 bit. Key IDs should be long enough (about 100 bit) to avoid dictionary attacks on hash values that are announced in the protocol CALL—note that potential collisions of key IDs are a minor issue, due to key confirmation. Thus, we end up with a total key record size of about 300 bit. This is a



(a) Small-world "beta" graph with rewiring parameter $\beta = 0.1$. Compromised nodes are drawn as red triangles.



(b) The number of uncompromised and compromised connections by the number of performed key exchanges.

Figure 4: Simulation on a small-world graph. Graph properties are: $|V| = 100$, $|E| = 200$, characteristic path length $L = 4.582$, clustering coefficient $C = 0.337$, diameter $D = 9$. The degree of compromisation is $\rho = 0.3$. There are $2\,415$ possible connections. After $2\,734$ simulated key exchanges, there are $2\,415$ uncompromised connections and 0 compromised connections.

rather optimistic calculation, but even using SHA-256 and 128-bit-IDs as well as five 24-bit-counters results in a key record size of only 504 bit. Given 1 GB[1] of memory reserved for keys, one can store more than 15.87 million key records of the bigger type (26.67M in the optimistic case). Bluetooth version 2.1 + EDR (transmission rate: 2169.6 kbit/s) allows for a theoretical throughput of about 4300 key records per second (or 7232 respectively).

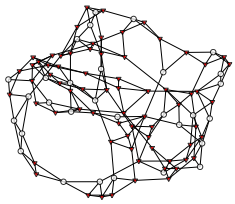
4 Heuristic Analysis

In order to evaluate the performance and security of SOKEN a simulation framework was developed. Because it allows the use of any undirected (connected) graph as the underlying social network, the framework allows us to evaluate SOKEN for various social network models.

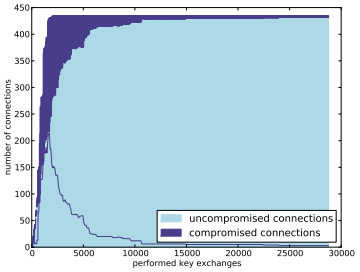
The simulation works as follows. An undirected graph is supplied. For each vertex a corresponding object that represents a participant is generated. The following procedure is then carried out until the simulation is aborted. A vertex is chosen uniformly at random. This vertex now initiates a group meeting. Each pair of attendees then executes the MEET procedure. (Keys received during this meeting are kept in a second keystore and only saved in the keyring $\mathcal{C}$ when the meeting is over.)

Group meetings work as follows. The initiator of a meeting is assigned an invitation budget which is chosen uniformly at random from $\{0, \ldots, 5\}$. He may invite as many of his neighbors until his budget is exhausted. (The cost of each invitation is 1.) Then,

[1] The value of 1 GB serves as an easy example to illustrate feasability of our approach. For comparison, modern mobile phones feature up to 16 Gigabytes of internal memory.



(a) Small-world "beta" graph with rewiring parameter $\beta = 0.1$. Compromised nodes are drawn as red triangles.



(b) The number of uncompromised and compromised connections by the number of performed key exchanges.

Figure 5: Simulation on a small-world graph with a high degree of compromisation. Graph properties are: $|V| = 100$, $|E| = 200$, characteristic path length $L = 4.842$, clustering coefficient $C = 0.382$, diameter $D = 10$. The degree of compromisation is $\rho = 0.7$. There are 435 possible connections. After $28\,782$ simulated key exchanges, there are 432 uncompromised connections and 3 compromised connections.

each neighbor invited gets assigned his own budget ($\in \{0, \ldots, 5\}$) and may invite as many neighbors of his as his budget allows. (Neighbors of neighbors are not allowed to bring additional guests.) If the initiator of a meeting is assigned a budget of 0, he may invite exactly one neighbor who doesn't get his own invitation budget.

The protocol MEET is implemented as described in Section 2. However, not all received keys are stored instantly. Keys received during a meeting are stored in a special keycache. This keycache is emptied after each meeting. Only keys whose key distribution paths are disjoint to the key distribution path of any other key in the keystore $\mathcal{C}$ are copied to $\mathcal{C}$. For keys with nondisjoint key distribution paths, only the key with the shorter key distribution path is kept. This measure has been taken to restrict memory consumption and to only capture the keys on which security relies. However, this optimisation is only applicable in a simulation environment, because SOKENs anonymity prevents the neccessary decisions being made.

Attacks can be carried out either by an adversary posing as a participant of the network or by an external adversary. An external adversary might eavesdrop on several key exchanges. This would make a number of exchanged key records known to him. Also, a compromised node could participate in the network as intended by the protocol, but also share all intermediary key material it is supposed to delete with the adversary.

We investigated the effect of such attacks and the resilience of various social network structures against this kind of passive attack. Because we deem a pure eavesdropping approach of any adversary improbable—the eavesdropping on many location-limited channels in parallel is cumbersome—we investigated the effect of multiple compromised parties among the regular network participants. We modeled an adversary orchestrating $\rho \cdot |V|$ compromised participants which are distributed evenly among uncompromised partici-

pants. (For $\rho \in [0, 1]$.) A key that gets stored in a keystore of a compromised participant is marked as compromised. This mark is never cleared. Let the shared keys between two uncompromised participants in their entirety be their *connection*—that is, a connection corresponds to exactly one edge in the Connectivity Graph of the network—then there are $\binom{n}{2}$ possible connections for n uncompromised participants. A connection is called compromised if it consists solely of compromised keys, and uncompromised if it contains at least one uncompromised key.

After each group meeting we determine the number of key exchanges performed so far and the number of compromised and uncompromised connections.

We used various graphs in our simulations. Figure 4 shows a small-world graph (generated with Watts' "beta" algorithm [Wat03]) with 100 nodes, a characteristic path length of 4.582 and a clustering coefficient of 0.337. The rate of compromisation is 0.3. The simulation results show a steep rise of the total number of connections. The number of compromised connections grows quickly but drops as quickly from a certain point on. After less than $3\,000$ key exchanges, all possible connections are made (there are $\binom{100-30}{2} = 2\,415$)—none are compromised.

Figure 5 shows a small-world graph with similar characteristics. The rate of compromisation is 0.7, however. Albeit taking longer than simulations with a lesser degree of compromisation, the simulation reaches a total of 3 compromised connections after less than $30\,000$ simulation steps.

References

[ACP04] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: smart trust for smart dust. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 206–215, Oct. 2004.

[AGLL94] Derek Atkins, Michael Graff, Arjen K. Lenstra, and Paul C. Leyland. The Magic Words are Squeamish Ossifrage. In Josef Pieprzyk and Reihaneh Safavi-Naini, editors, *Advances in Cryptology - ASIACRYPT '94*, pages 263–277. Springer, 1994.

[BSSW02] D. Balfanz, D. K Smetters, P. Stewart, and H. C Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, 2002.

[CHC+05] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. *Technical Report UCAM-CL-TR-617, University of Cambridge, Computer Laboratory*, 2005.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.

[Gar77] Martin Gardner. Mathematical Games: A New Kind of Cipher That Would Take Millions of Years to Break. *Scientific American*, 237:120–124, August 1977.

[GKLL09] Roxana Geambasu, Tadayoshi Kohno, Amit Levy, and Henry M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. In *Proc. of the 18th USENIX Security Symposium*, 2009.

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[Mil67] S. Milgram. The small world problem. *Psychology today*, 2(1):6067, 1967.

[NS78] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):999, 1978.

[Rab03] M.O. Rabin. Hyper-encryption by virtual satellite. *Science Center Research Lecture Series, December*, 2003.

[SCP+04] Jing Su, Alvin Chin, Anna Popivanova, Ashvin Goel, and Eyal de Lara. User Mobility for Opportunistic Ad-Hoc Networking. *Mobile Computing Systems and Applications, IEEE Workshop on*, 0:41–50, 2004.

[Sho94] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE*, volume 35, pages 124–124. Citeseer, 1994.

[SNS88] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: An authentication service for open network systems. In *Proc. Winter USENIX Conference*, pages 191–201. Citeseer, 1988.

[Wat03] D. J Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton Univ Pr, 2003.

ISBN 978-3-88579-286-4