Secure communication with secret sharing

in static computer networks with partition in mistrust parties

Johannes Viehmann
Fraunhofer Institut FOKUS (MOTION)
Kaiserin-Augusta-Allee 31
D-10589 Berlin, Germany
Johannes.Viehmann@Fokus.Fraunhofer.de

***Abstract* — This paper presents a solution to ensure arbitrarily secure communication in a large computer network by using secret sharing and multiple parties mistrusting each other instead of relying on some "trusted party" or a "web of trust". In contrast to other solutions that use a PKI and require asymmetric encryption, this concept can guarantee to provide secure communication even after any possible advance in cryptanalysis and even if unlimited calculation power was available to attack it. But this solution requires the computer network to have special properties. It is mainly intended to be used in the S-Network, a repository for reliable publications.**

Secret sharing; trust; mistrust; secure communication; PSMT

I. Basic Notation and Base Techniques

Let x and y be bit sequences. The concatenation of x and y prefixed with their identifiers and lengths is noted as x∘y.

The symmetric encryption of a bit sequence x with key K is notated as $E_K(x)$. The corresponding decryption is written as $D_K(E_K(x))$. Let P(x) be a function calculating a message authentication code (MAC) of a bit sequence x.

K, $E_K(x)$, $D_K(E_K(x))$, P(x) and x∘y are bit sequences. Messages are bit sequences, too.

A *secure channel* between Alice and Bob is a communication channel which allows exchanging messages between Alice and Bob in a finite time so that the secrecy, integrity and authenticity of the messages can be ensured and that the temporal order in which the delivered messages were sent by Alice can be reconstructed by Bob. There are provable secure solutions to keep perfect secrecy [14], but integrity, authenticity and the order can only be ensured with arbitrary high probability: Bit sequences passing tests for these could be guessed.

Secret sharing is a technology to split a secret x into a set of n pieces with the property that you need at least t pieces of the set to be able to reconstruct x from that subset. Any subset with less than the *threshold* t pieces does not reveal any information about x at all. There are several perfectly secure secret sharing systems known, for an example see [13]. The following notation will be used for the set of pieces of a secret sharing split:

$$Z_{n_t}(x)=\{T_{n_t 0}(x),\ \dots,\ T_{n_t n-1}(x)\}$$

The inverse operation will be noted as:

$$x=Z_{n_t}^{-1}(M)\ \mid M\subseteq Z_{n_t}(x)\ \wedge \#M\geq t$$

The concept introduced in this paper makes use of security technologies like secret sharing that do have a threshold up to which they are secure. To describe a unique security level for the entire system, a *security constant Ψ* is defined. Ψ is a natural number and it must be greater than two.

II. The Problem

A. Computer networks with strong security requirements

To enable secure communication in a computer network, any two participants should be able to establish a secure channel with each other. The required level of security may vary from application to application. Sometimes, long term security has to be guaranteed, which means, that the cryptographic concept should be secure and practically useable for the future – independent from any possible further technical development. For the following, high long term security requirements are assumed.

B. The difficulty to provide secure channels in big networks

It is possible to build an arbitrarily secure channel between any two participants Alice and Bob which has these strong long term security properties, but that requires that Alice and Bob share an exclusive secret in advance. Alice and Bob have to check their identities and exchange the secret manually.

In a very small network with only a few participants, it is possible to do such a manual procedure for all possible pairs of participants. But the effort grows quadratically with the number of participants. With several thousand or with several million participants, this is definitely not manageable.

III. State of the Art

A. Secure communication relying on a "trusted party"

"Trusted parties", sometimes called "trusted third parties", too, can be used to provide secure communication between any two participants in computer networks. The idea is, that all participants identify themselves only to the "trusted party". For the further usage of the "trusted party" there are different concepts:

1) *Inline usage* of a "trusted party": Each participant shares an exclusive key with the "trusted party" so that a secure channel can be built between each participant and the "trusted party". Messages between two simple participants Alice

Integrated Graduate Program H-C3 (IGP H-C3) of the Technische Universität Berlin (http://www.h-c3.org).

Presented at PST 2011 in Montreal, published in the conference proceedings (http://www.ieee.org),

and Bob are first send from Alice to the “trusted party” over a secure channel and then the “trusted party” forwards them to Bob over another secure channel. All messages have to pass the “trusted party”, which makes it likely that the central “rusted party” becomes a bottleneck.

2) *Online usage* of a “trusted party” as key server: The “trusted party” generates a session key for Alice and Bob so that they can build a direct secure channel between them. See [10] for a solution with this approach. With keys of constant length, this approach reduces the workload of the “trusted party”.

3) At least somehow *offline usage* of a “trusted party” or of a hierarchy of several “trusted parties” as certification authority (CA): See [7] for a description of such a public key infrastructure (PKI) and a discussion of the advantages in comparison with a key server. The most important difference for the strong long term security requirement is however, that asymmetric encryption (for example [12]) is required for this solution.

   It is possible that in the future all potential asymmetric algorithms can be broken in a relevant short time. For algorithms whose security depends on the assumed difficulty of calculating discrete logarithms or to do prime factorization for large numbers, a theoretical solution for breaking them with quantum computers in polynomial time has already been shown in [15]. For the prime factorization of small numbers, it has been demonstrated that the Shor algorithm really works [9].

   If potentially insecure functions are used for creating signatures on certificates, this may be another potential point to attack a PKI. See [16] for an attack that takes advantage of the MD5 cryptographic hash function that has been widely used on certificates, but which is not collision resistant.

   Public keys are typically used to encrypt and exchange symmetric session keys so that messages can then be exchanged with a more efficient symmetric cypher like AES [5]. This is called hybrid encryption. However, if the security of the symmetric cypher used for hybrid encryption might eventually be broken, this opens another point to attack. Recent advances in cryptanalysis [3] show that this threat should be taken serious.

   So in a typical PKI, there are at least three different potentially insecure algorithms that can be attacked independently. It is enough to break just one of these potential weaknesses to break the entire system's security.

No matter how “trusted parties” are used – the security of the communication depends on the fair and always correct behavior of the “trusted party”. Why should participants trust the “trusted party”? To control institutions that have so much power is difficult and maybe it is utopian or naive to believe that universal neutrality can at all be enforced in a big network that really matters.

B. *Secure communication relying on a "web of trust"*

To avoid the need to trust in some single party, the “web of trust” offers a decentralized alternative concept [4]. However, with this approach, it is not possible to achieve legal validity and it requires asymmetric encryption, too. Furthermore, the demands for the users are high as they have to decide whom to trust.

In general, it has also to be questioned whether trust is transitive at all.

C. *Secure communication with secret sharing*

Secret sharing can be used to avoid the need to trust a single party, too, by dividing the responsibility for trust related things between several parties. A typical application of secret sharing is to store a secret, for example a secret key.

But it is also possible to use secret sharing for “perfectly secure message transmission” (PSMT) over disjoint paths as shown in [6] (see also [8] and [11]). These solutions require a set of completely separated communication channels (called “wires”) between sender and addressee. But how these disjunct “wires” could be realized is not mentioned, neither how the identities could be checked nor how authentication could work.

In [1] a dynamic method to find separate wires is presented, but it provides only paths with disjunct edges, not with disjoint nodes. Therefore, it is not a solution for PSMT.

IV. Solution with Secret Sharing and Mistrust

A. *A static network with partition in mistrust parties*

The concept for secure communication presented in this paper requires an applicable legal framework and it requires the computer network in which the secure communication takes place to have the following properties:

The logical addresses of the logical systems within the network must be everlasting, absolute and unique. In the following, such an uniquely addressable logical system will be called an *S-Node*.

S-Nodes added to the network have to be kept accessible by their logical addresses. If an *S-Node* is not accessible because of some failure, it has to be repaired and restored within a finite time. Such a network may be called a *static network*.

For each S-Node there must be exactly one natural or juristic person responsible for it in a legal sense: the *S-Operator*. Each S-Node does also have an *owner*. If the S-Operator is not also the owner of his S-Node but only the administrator, the S-Operator must have a contract with the owner.

Let X be the set of all S-Operators in a static network. A *partition* of such a static network is the split of X into not empty disjunct subsets so that the union of all subsets is X. The subsets of a partition of a static network are called *parties*.

The solution presented in this paper requires a special partition of the static network so that any two S-Operators belonging to two different parties mistrust each other in a way that they will not cooperate for illegal and therefore potentially dangerous manipulations. Such a partition is called a *partition into mistrust parties*.

This mistrust between the parties can be established by a strict geographical, cultural and legal separation, by laws that prohibit certain forms of cooperation explicitly and by active measures to test the correct behavior of the S-Operators in the sense of these laws. Such a test can include fake proposals for building manipulative coalitions, for example. S-Operators have the duty to report illegal offers they get in a standardized fashion. Because any illegal offer could just be a fake for testing the correct reaction, not reporting them might be very risky.

MP is used as abbreviation for *mistrust party* in general. A certain MP is identified with an index i and noted as MP_i. If an S-Operator belongs to MP_i, all the S-Nodes he is responsible for belong to MP_i, too. $\#(MP_i)$ is the total number of S-Nodes belonging to mistrust party MP_i.

The general idea for the following solution for secure communication between two arbitrary S-Nodes is to split responsibilities among these mistrust parties.

B. Acquaintances, partisan forwarding

Two S-Nodes are called *acquaintances*, if messages can be exchanged between them over an arbitrary secure channel. Therefore the S-Operators of the acquaintances have to check the identities of each others S-Node's owner and they have to exchange the necessary communication data (including an exclusive secret key). This security critical manual operation is a high effort.

The S-Operators do also have to make sure that data can actually be transmitted between acquaintances in a finite time. Therefore, S-Operators of two S-Nodes that are acquaintances have to negotiate manually appropriate physical channels and they have to provide them to the S-Nodes. For example, one channel could be a direct microwave transmission and the Internet could be used as another single channel between the acquaintances.

Because of the high manual effort, an S-Node cannot have more than just a few acquaintances to be practicable.

Acquaintances do have high responsibility for each other. In order to split responsibilities between the mistrust parties, an arbitrary S-Node S_x must get for each mistrust party MP_i at least one S-Node belonging to MP_i as acquaintance. This ensures that the identity of the owner of S_x has to be verified for each different mistrust party at least by one S-Operator belonging to that MP whose S-Node becomes an acquaintance.

But to keep the manual effort on a reasonably low level, each S-Node should not require many more acquaintances than the total number of mistrust parties.

Only acquaintances may communicate directly with each other. If two S-Nodes are not acquaintances, a message can be exchanged between them if there is a series of pairwise acquaintances among them and if the message can be forwarded from one acquaintance to the next acquaintance in that series. Such an indirect connection is called a *forwarding*. The forwarding S-Nodes between the *sender* and the *addressee* are called *forwarders*.

For the solution presented in this paper, any two S-Nodes must be acquaintances or there must be a forwarding between them. In contrast to the direct communication with an acquaintance, the forwarding communication cannot take place over a secure channel because a sender and an addressee who are not acquaintances do not have an exclusive shared key with each other – they do not even know whether their pretended communication partner exists at all.

To make the communication between S-Nodes which are not acquaintances secure and reliable, there are additional requirements. For any two S-Nodes S_A and S_B belonging to the same MP_i, there must be a connection without any S-Node of all the other mistrust parties involved. This means, that if S_A and S_B are not acquaintances, there must be a forwarding between them so that all the forwarders belong to MP_i. Such a connection within a single MP is called *partisan forwarding*.

If the network structure within MP_i is like a single ring so that each S-Node belonging to MP_i has exactly two acquaintances in MP_i, there is always a partisan forwarding between any two S-Nodes belonging to MP_i if they are not acquaintances.

C. Partition-routing

For secure communication between any two S-Nodes S_A and S_B that are not acquaintances the following protocol for *partition-routing* may be used:

1) *Preparation*: Let x be the bit sequence to be transmitted. S_A creates a bit sequence x_p containing a random one-time key K_R, the encryption $E_{K_R}(x)$ and a message authentication code $P(K_R \circ x)$.

   $x_p = K_R \circ E_{K_R}(x) \circ P(K_R \circ x)$

   Let n be $n \in \mathbb{N} \wedge n \geqslant \Psi$. S_A builds the set of a secret sharing split:

   $Z_{n_\Psi}(x_p) = \{T_{n_\Psi 0}(x_p), \dots , T_{n_\Psi n-1}(x_p)\}$

   Let A_B be the address of the addressee S_B. Let H be additional required header data including some message number and the current time. S_A generates the n split messages τ_i:

   $\tau_i = A_B \circ H \circ T_{n_\Psi i}(x_p) \ \ \forall \ i \in \mathbb{N} | i < n$

2) *Separation*: S_A sends each τ_i over a secure channel to a different acquaintance of S_A not belonging to any of the mistrust parties S_A or S_B belong to. S_A may not send more than one piece of $Z_{n_\Psi}(x_p)$ into any MP.

3) *Check and forwarding*: Each forwarding S-Node S_f decrypts and checks messages m arriving over secure channels from its acquaintances.

   If m is from an acquaintance not belonging to the same MP as S_f, this acquaintance is the sender S_A. S_f generates an identity confirmation I_{Ai} containing the address of S_A, the name and additional identity data that was manually exchanged and verified when S_A and S_f became acquaintances. S_f adds I_{Ai} as proof of authenticity to the message m:

   $m = \tau_i \circ I_{Ai}$.

   Else if m is from an acquaintance belonging to the same MP as S_f, m must already contain an I_{Ai}.

   S_f must forward correct messages m for an addressee S_B according to these rules:

   3.1) If S_B is not an acquaintance of S_f, S_f forwards m over a secure channel to the next forwarder, who must be one of the acquaintances of S_f belonging to the same MP as S_f. The forwarder must be chosen so that the message gets closer to an acquaintance of S_B belonging to the same MP as S_f.

Continue with step 3 for the next forwarder.

3.2) Else if S_B is an acquaintance of S_f, S_f forwards m over a secure channel direct to S_B.

Continue with step 4.

4) *Check and collection*: The addressee S_B decrypts and checks messages arriving over secure channels from its acquaintances and extracts $T_{n_\Psi i}(x_p)$ from τ_i if possible. Correct arriving parts $T_{n_\Psi i}(x_p)$ and the according identity confirmations I_{Ai} are collected and stored together with the information from which MP they actually were forwarded.

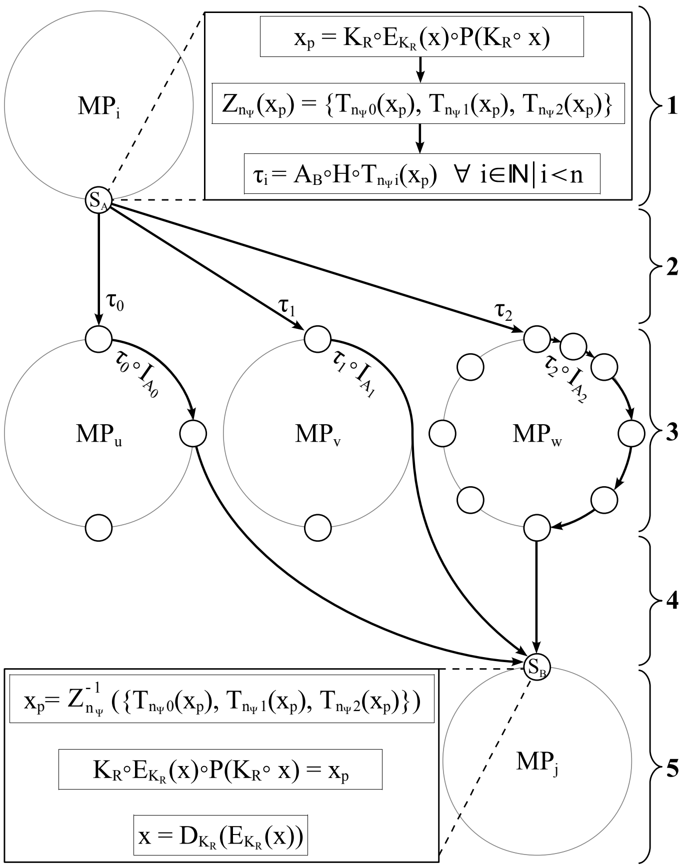

Figure 1. partition-routing with Ψ=n=3. MP_i may stand for the same MP as MP_j, but MP_u, MP_v and MP_w have to be distinct mistrust parties.

5) *Reconstruction and final check*: As soon as at least Ψ parts $T_{n_\Psi i}(x_p)$ of the set $Z_{n_\Psi}(x_p)$ arrived correct at the addressee S_B together with the confirming I_{Ai} from Ψ different mistrust parties, S_B can try to reconstruct x_p from that subset of $Z_{n_\Psi}(x_p)$.

The original data x can be decrypted with K_R:

$x = D_{K_R}(E_{K_R}(x))$.

The integrity can be checked with K_R, x and $P(K_R \circ x)$.

Only the sender S_A and the addressee S_B do get more than one piece of $Z_{n_\Psi}(x_p)$ if this protocol is followed properly: In step 2, all the parts $T_{n_\Psi i}(x_p)$ are distributed over secure channels to different mistrust parties. The forwarding of the loop in step 3.1 between an acquaintance of S_A and an acquaintance of S_B is a strictly partisan forwarding over secure channels. This means that all the parts $T_{n_\Psi i}(x_p)$ stay in exactly the MP they were sent to at step 2 until they reach an acquaintance of S_B. Only then, at step 3.2, all the parts are send to the same MP, but they are directly send over secure channels to the addressee S_B.

To reconstruct x_p from a subset of $Z_{n_\Psi}(x_p)$, at least t=Ψ parts of $Z_{n_\Psi}(x_p)$ are required. Any attack to get x_p and therefore any manipulation that is more sophisticated than just trying to guess an entire valid bit sequence must affect at least Ψ forwarding S-Nodes in Ψ different mistrust parties.

The identity confirmation I_{Ai} as proof of authenticity has to be identical from at least Ψ different mistrust parties, too. To cheat requires again that at least Ψ S-Nodes in Ψ different mistrust parties behave incorrect.

D. Optimization

With the simple ring like network structure shown so far, each S-Node has exactly two acquaintances belonging to the same MP and that is theoretically enough because the required partisan forwarding is possible with that solution. But this is not yet a practicable solution, because there are two major weaknesses:

1) The *efficiency* is unusably low. Partisan forwarding may need great many S-Nodes as forwarders. In the worst case, a message has to be forwarded by 50% of the $\#(MP_i)$ S-Nodes that belong to MP_i. With thousands or millions of S-Nodes, this would be terribly slow because the messages are not just forwarded – they have to be decrypted, checked and encrypted with another key. On average each S-Node would have to forward about 25% of all the messages exchanged by forwarding through its MP which can result in an extremely high workload, too.
2) The total system *robustness* would be very low. If only two S-Nodes belonging to the same MP are temporary not reachable for their acquaintances, the entire ring like network structure can break into two separate segments R and Q so that any partisan forwarding between an S-Node in R and another S-Node in Q would fail.

Robustness against failures in a communication network can be increased by mashing up the network tighter with additional redundant connection possibilities so that alternative routes can be chosen in case of failures [2].

By increasing the number of acquaintances within the same MP per S-Node, alternative routes for the partisan forwarding can be created. But more acquaintances imply also a higher manual effort.

With just a few more carefully chosen acquaintances for each S-Node and with a fitting routing concept, a good robustness can be achieved. By doing so, the length of the most efficient partisan forwarding between any two S-Nodes belonging to the same MP can be reduced to a practical value, too. The solution presented here is completely decentralized.

Requirements, definitions and strategic objectives

The following optimization requires the static address of an S-Node to consist of two independent components – one identifying the mistrust party MP_i the S-Node belongs to and the other identifying the S-Node within MP_i. The last is called the *Intra-MP-Address*. The Intra-MP-Address must be a natural number and it must be unique within its mistrust party. The S-Nodes belonging to the same MP_i can be sorted by their Intra-MP-Addresses.

For the optimization, for each S-Node S_x belonging to MP_i, two acquaintances belonging to the same MP_i are chosen according to the following rules:

1) The S-Node with the biggest Intra-MP-Address in MP_i smaller than the Intra-MP-Address of S_x becomes an acquaintance of S_x, if such an S-Node exists.
2) The S-Node with the smallest Intra-MP-Address in MP_i bigger than the Intra-MP-Address of S_x becomes an acquaintance of S_x, if such an S-Node exists.
3) Additionally, the S-Node belonging to MP_i with the smallest Intra-MP-Address in MP_i and the S-Node belonging to MP_i with the biggest Intra-MP-Address in MP_i become acquaintances.

This results again in a ring like network structure per MP, but the S-Nodes on that ring are now sorted by their Intra-MP-Address.

The *ring-distance* $R(S_A, S_B)$ between two S-Nodes S_A and S_B belonging to the same MP_i is the number of S-Nodes on the sorted ring that are between S_A and S_B in the shorter direction.

The ring-distance is useful to define an objective for the optimization of the partisan forwarding with adding additional acquaintances in the same MP:

In the partisan forwarding process of a message between two arbitrary S-Nodes S_A and S_B belonging to the same MP_i, it should be possible to reduce the ring-distance to S_B at each forwarding step from S_{old} to S_{new} according to this formula:

$$R(S_{new}, S_B) \leq R(S_{old}, S_B) - \frac{R(S_{old}, S_B)}{d} \quad \text{with } d \in \mathbb{N} \wedge d > 1$$

If the constant divisor d is 2 for example, the new ring-distance should at least be reduced by 50% of the old ring-distance at each optimized partisan forwarding step.

Let F_i be the number of S-Nodes required as forwarders between S_A and S_B in an optimized partisan forwarding in MP_i. F_i would then be logarithmic with the number of S-Nodes belonging to MP_i:

$$F_i \leq (d-1) * \lceil \log_d(\#(MP_i)) \rceil$$

Let A_i be the number of acquaintances each S-Node S_x needs in his own MP to provide such an efficient partisan forwarding. If the acquaintances are choosen well distributed, the upper bound of A_i for this optimization can be:

$$A_i \leq 2 * \lceil \log_d(\#(MP_i)) \rceil$$

For d=2, F_i becomes minimal, but A_i becomes maximal, so the most acquaintances per S-Node will be required. Because making many acquaintances means a high manual effort, it probably makes sense to chose a higher d and to accept slightly longer routes in the partisan forwarding.

In theory, acquaintances are perfectly distributed if each S-Node S_x belonging to MP_i does always have exactly those S-Nodes belonging to the same MP_i as acquaintances that have a ring-distance of $(d^f - 1)$ with $f \in \mathbb{N} \wedge f < \lceil \log_d(\#(MP_i)) \rceil$ to S_x. Because the ring-distances might change whenever a new S-Node is inserted into MP_i and making new acquaintances has a high manual effort, for this optimization an approximation to the perfect distribution with enduring well-chosen acquaintances is the best solution.

It is not enough that efficient routes with no more than F_i forwarders for the optimized partisan forwarding just exist: With the help of the Intra-MP-Address of the addressee S_B, any S-Node must actually be able to chose the best acquaintance a message should be forwarded to in order to bring it closer to S_B on the optimal route.

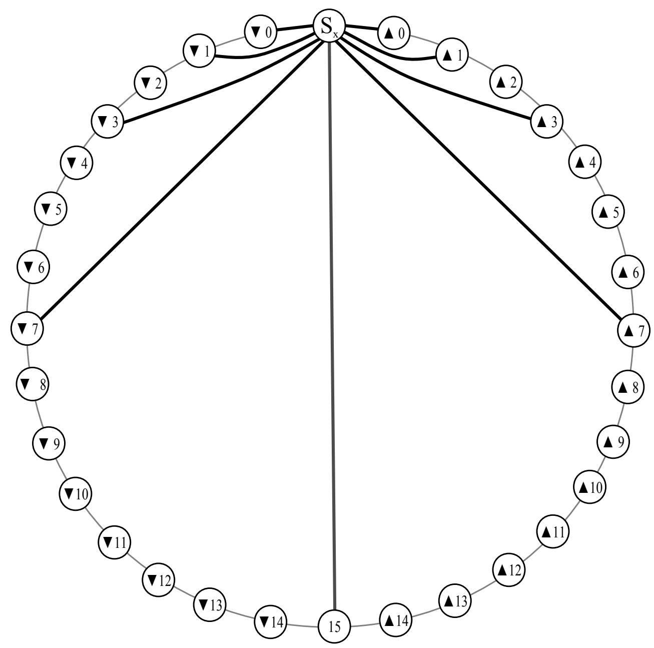

Figure 2: The optimal distributed acquaintances of S-Node S_x in MP_i with d=2. The numbers in the small circles that represent the other S-Nodes in MP_i indicate their ring-distances to S_x.

Procedure to add S-Nodes and to make acquaintances in a MP

This procedure creates and inserts new S-Nodes with their Intra-MP-Addresses into MP_i and makes the optimized acquaintances within the same MP_i.

In advance, two constants $d \in \mathbb{N} \wedge d > 1$ and $z \in \mathbb{N}$ have to be defined. Intra-MP-Addresses might have any values of natural numbers between 0 and d^z, so d^z should be much bigger than the potential number of S-Nodes a single MP_i might actually ever have.

Let α be a variable for the current Intra-MP-Address and let r be an integer to count the rounds, which is initialized with 1.

1) *Initialization*: The first S-Node in MP_i gets the Intra-MP-Address $\alpha=0$. No acquaintances in MP_i have to be made.
2) *Increment* of α: As long as an S-Node with the Intra-MP-Address α exists already in MP_i, α is incremented with $d^z \div d^r$.
3) *Check* for new round: If $\alpha \geq d^z$, then r is incremented by one and α is set to $\alpha = d^z \div d^r$.
4) *Insertion*: A new S-Node $S_{i\alpha}$ with the Intra-MP-Address α is inserted.
5) *Make acquaintances*: $S_{i\alpha}$ should get for each $f \in \mathbb{N} | f > 0 \wedge f \leqslant r$ an acquaintance with the Intra-MP-Address $\beta_+ = (\alpha + d^f)$ modulo d^z and an acquaintance with the Intra-MP-Address $\beta\_ = (\alpha - d^f)$ modulo d^z .

   The following sub-steps have to be done for each optimal address β_+ and for each optimal address β_-. The notation $\beta_\pm$ will be used to express that.

   5.1) If there is already an S-Node with the Intra-MP-Address $\beta_\pm$ in MP_i, then this S-Node $S_{i\beta\pm}$ becomes a *final optimal acquaintance* of $S_{i\alpha}$.

   5.2) Else there is not yet an S-Node with the Intra-MP-Address $\beta_\pm$ in MP_i.

   Let the *address-distance* $\Delta(\varphi,\chi)$ between two Intra-MP-Addresses φ and χ be:

   $$\Delta(\varphi, \chi) = \min(\{abs(\varphi - (\chi + p * d^z)) \,|\, p \in \mathbb{Z}\})$$

   Let W be the set of all S-Nodes in MP_i that have a smaller address-distance to α than $\Delta(\beta_\pm,\alpha)$.

   If W is not empty, then the S-Node $S_{i\gamma}$ whose Intra-MP-Address γ has the smallest $\Delta(\gamma,\beta_\pm)$ of all the S-Nodes in W becomes a *preliminary suboptimal acquaintance* of $S_{i\alpha}$.

   Else if $r = 1$ then the S-Node with the Intra-MP-Address 0 becomes a *preliminary suboptimal acquaintance* of $S_{i\alpha}$.

   For the next S-Node to be inserted, continue with step 2.

Note: At the end of each round when r is incremented, all the perfectly distributed acquaintances exist. Whenever an already existing S-Node $S_{i\beta}$ becomes a final optimal acquaintance of a new S-Node $S_{i\alpha}$, some preliminary suboptimal acquaintances of $S_{i\beta}$ might become superfluous.

The average total number of suboptimal and optimal acquaintances made per S-Node in its own MP_i is less than: $1.5 * A_i = 3 * \lceil \log_d(\#(MP_i)) \rceil$.

Foresighted partisan forwarding

In the process of partisan forwarding each forwarder S-Node being not an acquaintance of the addressee S_B has to identify the acquaintance that would be the next optimal forwarder. In a network constructed the way shown before, that is the acquaintance with the Intra-MP-Address having the lowest address-distance to the Intra-MP-Address of S_B.

If some S-Node S_F would be the next optimal forwarder, but the current forwarder S_E cannot reach S_F, alternative routes may be tried until S_F is restored and reachable again. Alternative routes are not necessarily less efficient. To find the best alternative route is however more difficult: the address-distances have to be checked further ahead.

Let X be a set of S-Nodes that belong to MP_i. Let B(X) be the set of those S-Nodes belonging to the same MP_i which have at least one acquaintance in set X.

For optimal routing, S_E has to choose the S-Node in $B(B(\{S_E\})\backslash S_F)\backslash S_E$ as next forwarder that has the Intra-MP-Address with the minimal address-distance to the Intra-MP-Address of S_B.

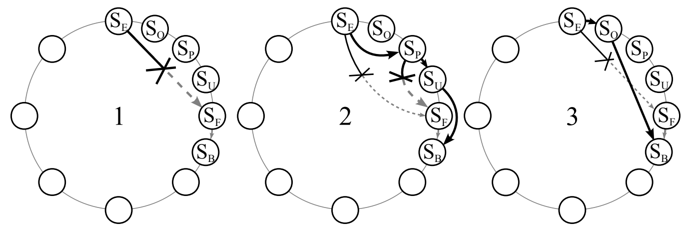

Figure 3. If S_F is not reachable (1), S_E has to find an alternative route. Choosing the acquaintance S_P which is the next closest to the addressee S_B (2) is less efficient than looking ahead and choosing S_O as the next forwarder (3).

Let S_U be an acquaintance of the addressee S_B. If the addressee S_B is not reachable for S_U, other acquaintances of S_B may be tried as final forwarders. Each S-Node in $B(\{S_B\})\backslash S_U$ that has not yet been tried can be chosen as a preliminary target on an alternative partisan forwarding route. The number of alternative acquaintances for each S-Node in MP_i is between $r-1$ and $r*2-1$ with $r = \lceil \log_d(\#(MP_i)) \rceil$.

Whenever an S-Node is not reachable and an alternative route is tried, this has to be logged in the message's header to make sure that no message circles around in an endless loop.

Acquaintances in foreign mistrust parties

In the protocol for partition-routing, between the first and the last forwarder only partisan forwarding is used to deliver each split message from the sender S_A to the addressee S_B. For an efficient routing, it is essential to find an acquaintance of S_B belonging to the MP in which the entire partisan forwarding takes place so that it can be used as preliminary target in the optimized partisan forwarding process.

Therefore, exactly those S-Nodes in different mistrust parties that have the same Intra-MP-Address should become pairwise acquaintances.

If in any MP_i there is an S-Node $S_{i\chi}$ with the Intra-MP-Address χ, but in another MP_j there is not yet an S-Node having the same Intra-MP-Address χ, $S_{i\chi}$ must get some suboptimal preliminary acquaintance in MP_j because each S-Node must have at least one acquaintance in each MP.

Let $S_{j\varphi}$ be the S-Node in MP_j having the greatest Intra-MP-Address smaller than χ. Then $S_{j\varphi}$ becomes the suboptimal preliminary acquaintance of $S_{i\chi}$ in MP_j.

If later an S-Node $S_{j\chi}$ is added to MP_j, then $S_{j\chi}$ becomes the optimal acquaintance of $S_{i\chi}$ in MP_j. The suboptimal preliminary acquaintance $S_{j\varphi}$ becomes superfluous for $S_{i\chi}$.

If an S-Node $S_{i\chi}$ has already a suboptimal preliminary acquaintance $S_{j\varphi}$ in MP_j, it would principally be possible to create a better placed suboptimal preliminary acquaintance as soon as a new S-Node $S_{j\nu}$ is inserted in the same MP_j having an Intra-MP-Address ν that is bigger than φ and smaller than χ. That would lead to shorter forwarding routes which are easier to find, but the additional manual effort to make acquaintances is probably too high.

Therefore, if a suboptimal preliminary acquaintance $S_{j\varphi}$ for $S_{i\chi}$ already exists, it seems to be better to create only one more acquaintance for $S_{i\chi}$ in MP_j, which must be the optimal acquaintance $S_{j\chi}$.

Protocol for optimized partition-routing

Let $S_{i\alpha}$ be the sender belonging to MP_i. Let $S_{j\beta}$ be the addressee belonging to MP_j and having the Intra-MP-Address β.

The following protocol has to be repeated for each split message of the partition-routing protocol. It delivers such a message m from $S_{i\alpha}$ to $S_{j\beta}$. All the forwarders must belong to the same MP. Let MP_v be that MP. Let S_{v*} be a variable for an S-Node belonging to MP_v.

1) *Check for common acquaintance*: $S_{i\alpha}$ sends m to an acquaintance S_{v*} belonging to MP_v. If S_{v*} is also an acquaintance of $S_{j\beta}$, S_{v*} can forward m directly to $S_{j\beta}$ and the protocol ends.
2) *Route to optimal acquaintance*: With the foresighted partisan forwarding, the S-Nodes in MP_v try to deliver m to an S-Node $S_{v\beta}$ belonging to MP_v and having the same Intra-MP-Address β as $S_{j\beta}$. If the S-Node $S_{v\beta}$ exists and can be reached, m can be forwarded in a single step from $S_{v\beta}$ and the protocol ends.

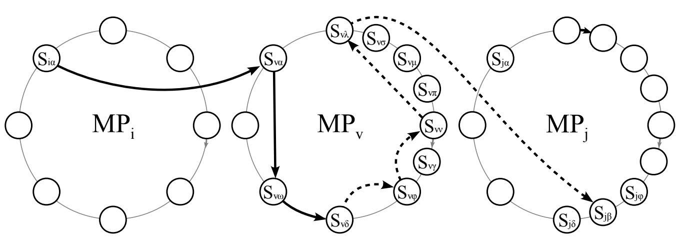

Figure 4. Searching a suboptimal preliminary acquaintance of $S_{j\beta}$ in MP_v: The S-Nodes $S_{v\gamma}$, $S_{v\pi}$, $S_{v\mu}$ and $S_{v\sigma}$ have been added at later rounds – after checking $S_{v\varphi}$ and $S_{v\nu}$, it is clear that they cannot be acquaintances of $S_{j\beta}$ so they may be skipped in the search process of the optimized partition-routing protocol.

3) *Go to start point for alternative search loop*: S_{v*} is set to the S-Node having the biggest Intra-MP-Address smaller than β in MP_v.

   If the foresighted partisan forwarding did not end at S_{v*}, but at $S_{v\chi}$, m must be send now to S_{v*}. This should always be possible in a single forwarding step because S_{v*} and $S_{v\chi}$ are at least acquaintances.
4) *Try to reach addressee*: If S_{v*} is an acquaintance of $S_{j\beta}$, m is forwarded to $S_{j\beta}$. End of the protocol.
5) *Check if search failed*: If S_{v*} has an Intra-MP-Address null, there is no acquaintance of $S_{j\beta}$ in MP_v. End of the protocol.
6) *Forward to next possible acquaintance*: For any S-Node $S_{x\chi}$ having the Intra-MP-Address χ let $\Phi(S_{x\chi})$ be the smallest natural number bigger null for that the equation χ modulo $d^{(z-\Phi(S_{x\chi}))} = 0$ holds. Note that $\Phi(S_{x\chi})$ is identical with the round r in which $S_{x\chi}$ was created.

   Let $L(S_{v*})$ be a subset of $B(S_{v*})$ containing only those acquaintances $S_{v\chi}$ that have a $\Phi(S_{v\chi})$ less or equal to $\Phi(S_{v*})$.

   S_{v*} forwards m to the S-Node of $L(S_{v*})$ having the biggest Intra-MP-Address that is smaller than the Intra-MP-Address of S_{v*}.

   That S-Node becomes the new S_{v*}.

   Continue with step 4.

Let F be the maximum number of forwarders required for the partition-routing of a message strictly split over n mistrust parties. With the optimized protocol, F is limited by the following formula:

$$F \leqslant \sum_{i=0}^{n-1} (2*F_i+3) = \sum_{i=0}^{n-1} (2*(d-1)*\lceil \log_d(\#(MP_i)) \rceil + 3)$$

To make communication more robust, the number of acquaintances in other mistrust parties could be increased.

But if the number of mistrust parties is bigger than the security threshold Ψ, in case of any disturbance in some MP_k it would be possible to avoid MP_k completely and choose another MP instead to deliver a split message. Or if for the Z_{n_Ψ} n is chosen bigger than Ψ, in up to $n-\Psi$ different mistrust parties there may be failures and the communication still works.

Also, secure connections between two acquaintances $S_{k\chi}$ and $S_{i\chi}$ belonging to two different mistrust parties are only used for communication having at least one of the S-Nodes $S_{k\chi}$ and $S_{i\chi}$ as sender or addressee. If the direct connection between $S_{k\chi}$ and $S_{i\chi}$ is disrupted, the effect of this failure is rather limited. Only those messages that have $S_{k\chi}$ or $S_{i\chi}$ as sender or addressee are affected.

Therefore, additional redundancy seems to be superfluous for acquaintances in foreign mistrust parties.

Let q be the number of mistrust parties. Any S-Node will not need more than $q-1$ acquaintances in all the other mistrust parties together.

Per S-Node belonging to MP_i, this leads to a total number TA_i of required optimal acquaintances according to the following formula:

$$TA_i \leqslant A_i + q - 1 = 2*\lceil \log_d(\#(MP_i)) \rceil + q - 1$$

With the shown procedures, in average additional SA_i suboptimal preliminary acquaintances will be created per S-Node:

$$SA_i \sim \lceil \log_d(\#(MP_i)) \rceil + (q-1) \div 2$$

V. *Possibilities*

The solution presented so far offers only secure communication between S-Nodes which should always be online. Clients that are typically often offline cannot be a part of a static network. But the human beings and their client systems using the static network should be able to communicate secure and reliable with any S-Node, too. There are several ways to do this:

An S-Node could work as a *proxy* server for his owner. The owner does only have to be able to communicate directly with his own S-Node over a secure channel. This S-Node can forward messages which should be exchange with other S-Nodes – using the partition-routing protocol for those other S-Nodes that are not acquaintances. For users having their own S-Node and who trust in its reliability, this is the preferred solution.

Of course, users could also exchange for each mistrust party MP_i the required information for direct communication over a secure channel with at least one responsible S-Operator of an S-Node belonging to MP_i. Then, they could themselves start the partition-routing protocol. The advantage would be that the user does not need his own proxy S-Node. A failure of such a single S-Node could not hinder the user to communicate with other S-Nodes. The disadvantage would be the higher manual effort.

A. Application

If a computer network is anyway static and if a partition in mistrust parties is required for other reasons than secure communication, too, then the security concept presented here does not produce much additional effort. The S-Network, a trustworthy repository currently developed at Fraunhofer FOKUS, is a good candidate for this concept: The S-Network combines secure long term data storage and preservation in a computer network with non-repudiation and an international applicable legal validity. For the future, the S-Network must be guaranteed to be secure even after any possible technical advance.

The S-Network uses mistrust parties to store backup copies in a distributed way and it requires secure communication between the systems storing the backup copies. With the concept presented in this paper, exactly the security level that is reached for data preservation can be guaranteed for the required message exchange. The same provable secure base technologies like secret sharing can be used, and of course the same mistrust parties, too.

The solution presented here was already successfully implemented in a prototype of the S-Network.

VI. CONCLUSION

This paper presents a practical concept for secure communication in a large computer network without a “trusted party” or a “web of trust” and without relying on assumptions of the complexity theory. Unlike in previous PSMT proposals, a realistic concept to actually create communications paths with disjunct sets of nodes is provided.

Depending on the choice of algorithms used to build secure channels between acquaintances, unlimited calculating power does not help to successfully break the security of this solution. That makes this solution applicable where ever a strong long term security concept is required.

Perfect security is not guaranteed: An attacker could randomly generate a message that passes the integrity tests. The likelihood that a valid message is guessed can be reduced by expanding the MAC.

The security also depends on the choice of Ψ: A coordinated manipulation involving at least Ψ S-Nodes in Ψ different mistrust parties can break the security concept. Increasing Ψ and the number of mistrust parties probably only makes sense up to a certain degree. It is really decisive to prevent manipulative cooperation among the mistrust parties.

For a trustworthy repository that has to guarantee secure long term serviceability like the S-Network, the solution presented here seems to be a good choice.

REFERENCES

[1] Amitabha Bagchi, Amitabh Chaudhary, Michael T. Goodrich, Shouhuai Xu: Constructing Disjoint Paths for Secure Communication; Lecture Notes in Computer Science, 2003, Volume 2848/2003 pp. 181-195; Springer Verlag Berlin 2003

[2] Paul Baran: On Distributed Communications Networks; RAND Corporation Santa Monica 1962; http://www.rand.org/pubs/papers/P2626 (2010-01-25)

[3] Alex Biryukov, Dmitry Khovratovich: Related-key Cryptanalysis of the Full AES-192 and AES-256; Cryptology ePrint Archive 2009; http://eprint.iacr.org/2009/317

[4] J. Callas et al.: RFC 4880: OpenPGP Message Format; The Internet Society 2007; http://tools.ietf.org/html/rfc4880 (2011-02-02)

[5] Joan Daemen, Vincent Rijmen: The design of Rijndael: AES - the advanced encryption standard; Springer Verlag Berlin 2002; ISBN 3-540-42580-2

[6] D. Dolev, C. Dwork, O. Waarts, M. Yung: Perfectly secure message transmission; 31st Annual Symposium on Foundations of Computer Science (FOCS 1990) 1990; http://www.computer.org/portal/web/csdl/doi/10.1109/FSCS.1990.89522 (2011-02-21)

[7] Niels Ferguson, Bruce Schneier, Tadayoshi Kohno: Cryptography Enineering; Wiley Publishing, Inc. Indianapolis 2010; ISBN: 978-0-470-47424-2

[8] Matthias Fitzi, Matthew Franklin, Juan Garay and S. Harsha Vardhan: Towards Optimal and Efficient Perfectly Secure Message Transmission; Lecture Notes in Computer Science, 2007, Volume 4392/2007 pp. 311-322; Springer Verlag Berlin 2007

[9] IBM Research Division; IBM's Test-Tube Quantum Computer Makes History; First Demonstration Of Shor's Historic Factoring Algorithm; ScienceDaily 2001; http://www.sciencedaily.com/releases/2001/12/011220081620.htm

[10] J. Kohl, C. Neuman: RFC 1510: The Kerberos Network Authentication Service; Massachusetts Institute of Technology 1993; http://www.ietf.org/rfc/rfc1510.txt (2011-02-02)

[11] Kaoru Kurosawa, Kazuhiro Suzuki: Truly Efficient 2-Round Perfectly Secure Message Transmission Scheme; IEEE Transactions on Information Theory, v.55 n.11 pp. 5223 - 5232; 2009

[12] RSA Laboratories: PKCS #1 v2.1: RSA Cryptography Standard; RSA Security Inc. 2002; ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf (2011-02-02)

[13] Adi Shamir: How to share a secret; Communications of the ACM Volume 22 Issue 11 pp. 612-613; ACM New York 1979

[14] C. E. Shannon: Communication theory of secrecy systems; Bell System Technical Journal 28 pp. 656 - 715; Bell Labs 1949

[15] Peter W. Shor: Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer; Bell Labs 1994; http://arxiv.org/abs/quant-ph/9508027v2

[16] Alexander Sotirov et al.: MD5 considered harmful today: Creating a rogue CA certificate; 25th Annual Chaos Communication Congress in Berlin 2008; http://www.win.tue.nl/hashclash/rogue-ca/