Threshold Things That Think: Authorisation for Resharing

Roel Peeters*, Markulf Kohlweiss, and Bart Preneel

K.U.LEUVEN, ESAT/SCD/COSIC and IBBT
Kasteelpark Arenberg 10, 3001 Leuven - Heverlee, Belgium
{firstname.lastname}@esat.kuleuven.be

Abstract

As we are evolving towards ubiquitous computing, users carry an increasing number of mobile devices with sensitive information. The security of this information can be protected using threshold cryptography, in which secret computations are shared between multiple devices. Threshold cryptography can be made more robust by resharing protocols, which allow recovery from partial compromises. This paper introduces user-friendly and secure protocols for the authorisation of resharing protocols. We present both automatic and manual protocols, utilising a group manual authentication protocol to add a new device. We analyse the security of these protocols: our analysis considers permanent and temporary compromises, denial of service attacks and manual authentications errors of the user.

1 Introduction

The arrival of ubiquitous computing results in users spreading their sensitive personal information across more and more "Things That Think" [1]. Things That Think are mobile devices with computational power and storage capabilities. The responsibility to protect information is shifting towards the end user, for whom convenience is often more important than security.

Desmedt et al. [4] have proposed a new approach for this setting based on threshold cryptography. A private key is shared among personal devices that are frequently in the user's proximity and able to interact with each other. The corresponding public key is known to all devices. If at least the threshold number of devices cooperate, they can jointly sign and/or decrypt data. The advantages of deploying a threshold cryptography scheme are twofold: a user does not need all his personal devices (e.g. empty battery, device left at home) to use the private key and an adversary does not gain any knowledge of the private key when he does not compromise the threshold number of devices.

* Roel Peeters is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

A mechanism for resharing the private key is needed. Resharing allows to change the set of devices, hence revoke decryption/signature rights, and forces an adversary to break the scheme within a certain time frame. Different schemes for secret resharing are described by Desmedt et al. [5], Frankel et al. [6] and Wong et al. [15].

In the literature, little attention has been paid to the problem of authorisation for resharing. Proper authorisation is necessary to prevent an adversary from altering the set of devices part of the threshold secret sharing in such a way that he would be able to break the scheme. Moreover authorisation should not allow the adversary to succeed in a *Denial of Service* (DoS) attack and prevent the genuine user from signing and/or decrypting. We are specifically looking at DoS attacks. This is relevant because mobile devices are constrained in resources, e.g. battery powered, and considered as critical infrastructure for the user.

We consider the following problem: designing a secure and user-friendly [2] protocol to authorise resharing. We solve this problem by introducing two ways to authorise resharing: automatic and manual.

Authentication for resharing is related to voting algorithms in distributed systems, such as the algorithms proposed by Castro [3] and Hardekopf [9]. The mobile devices need to decide if resharing is authorised or not.

This paper first gives an overview of the assumptions. It will then go on to describing the protocols for authorisation for resharing. The security of these protocols will be discussed in section 4.

2 Assumptions

2.1 Devices

Typically mobile devices have very different capabilities. Some devices have a *Graphical User Interface* (GUI), others do not. Some devices can store lots of data, while others have very limited storage capabilities. We assume that not all devices have a clock and no reliable global clock is available.

The user has n devices, all in possession of the public key and a share of the corresponding (k, n)-secret shared private key, where k is the threshold. Each device that participates to this secret sharing knows the public keys of the other devices in this secret sharing. These public keys are either stored by the device itself, or this device has constructed personal certificates [8]. A personal certificate contains the public key of another device in the threshold secret sharing. The personal certificates are broadcasted for devices with greater storage capabilities to store. After resharing, additional personal certificates are created and broadcasted.

2.2 Communication

Because in normal operation the distance between these devices is small and communication is wireless, we assume the broadcast model. We assume that all active devices carried by the user receive messages at the same time and that either all or no devices receive messages.

To set up private communication between two devices, a private channel needs to be constructed by deploying encryption over the broadcast channel. We assume that all devices in the secret sharing scheme know each other's public keys or can ask more powerful devices for their personal certificates in order to set up private channels.

2.3 Adversary

The adversary can compromise devices. A device is compromised if an adversary learns its key share. We consider two types of compromised devices: passively and actively compromised devices. Passively compromised devices are not controlled by the adversary, the adversary only knows the key shares of these devices. An example of a passively compromised device is a device to which an adversary had physical access to for a brief period of time, e.g., lunch break, and succeeded in reading out the key share. Actively compromised devices are controlled by the adversary. An example of a actively compromised device is a device on which an adversary succeeds in installing malware on. The number of devices compromised by the adversary is $t = t_p + t_a$.

The adversary can play two games: he can try to break the scheme or render the scheme useless to the genuine user. The threshold secret sharing scheme is broken if an adversary has compromised k devices. The adversary succeeds in a DoS attack if less than k devices follow the protocol. DoS attacks are even facilitated by not carrying around all devices and by devices that run out of battery power, since the adversary does not need to compromise these devices in order to get them not to follow the protocol.

We distinguish two kinds of DoS attacks, temporary and permanent DoS attacks. The adversary can try to reduce the number of devices that follow the protocol, by draining the batteries of the mobile devices. To drain the batteries, the adversary sends messages that trigger the devices to do some calculations and send other messages. If less than k uncompromised devices remain, the adversary succeeded in a temporary DoS attack. To protect against temporary DoS attacks the number of calculations and bits sent, during the time where a malicious party cannot be identified, needs to be minimised. A permanent DoS attack prevents a user from using any threshold functionality, even when the user has all his devices with fully charged batteries with him. This is the case we the adversary succeeds in compromising $(n - k + 1)$ devices.

We aim to protect against an adversary breaking the scheme and permanent DoS attacks. This results in two conditions on the number t of devices that we allow an adversary to compromise:

$$t < k \tag{1}$$

$$t \leq n - k \ . \tag{2}$$

3 Resharing

Resharing key material is essential for security. On the one hand resharing is a means to revoke decryption/signature rights of devices, for example, devices no longer in possession of the user. On the other hand resharing limits the timeframe in which an adversary needs to compromise enough devices to break the scheme. Resharing also allow the user to add new devices to the threshold secret sharing.

The key share of a device is updated after a successful resharing. The adversary does not know the new key shares of passively compromised devices. The new key shares of actively compromised devices are known to the adversary. After resharing, only passively compromised devices are no longer compromised:

$$t'_p = 0 \ . \tag{3}$$

We introduce two ways to authorise resharing: automatic and manual. Automatic authorisation for resharing is used for periodic updates. Manual authorisation for resharing requires user interaction. Manual authorisation for resharing is required if the user wants to alter the number of participants. After successful authorisation, resharing will take place. Each device checks if resharing is authorised, and trigger the resharing protocol.

3.1 Automatic authorisation

To limit the timeframe in which an adversary needs to compromise enough devices, the shares of the private key will be updated periodically. Since no reliable global clock exists, we will use the number of performed decryptions/signatures as an approximation for time. Each device i has a local counter c_i, counting the number of performed decryptions/signatures since the last (re)sharing of the private key. We define c as the number of decryptions/signatures after which resharing is recommended. The number of decryptions/signatures after which resharing is necessary is $C > c$. After successful resharing all devices set their local counter $c_i = 0$.

If the local counter of one device reaches c, this device requests automatic authorisation for resharing. If this request is denied, the device augments c with a step Δ, $c = c + \Delta$. If the local counter of one device reaches C, resharing is necessary. In this case several attempts to get automatic authorisation for resharing have failed; the user is alerted by the GUI-enabled devices that manual authorisation for resharing is needed.

Resharing is only authorised if all n devices are present and at least k devices agree to reshare. A device agrees to reshare if its local counter exceeds c', as defined in Eq. (4):

$$c' = \left\lfloor \frac{k-1}{n-1} \cdot c \right\rfloor \ . \tag{4}$$

The value c' is the expected value of a device's local counter if only the minimum number of devices (k) participate in every decryption/signature and the requesting device is always one of these.

Replay of the request and/or granting of the request for automatic authorisation needs to be prevented. This is prevented by the introduction of nonces. Fig. 1 shows the protocol for automatic authorisation.

1. Device j ($c_j > c$) broadcasts a session identifier.

$$broadcast(sess_{id}) \ .$$

2. Each device i broadcasts a nonce.

$$broadcast(sess_{id}, i, nonce_i) \ .$$

3. After receiving all other $(n-1)$ nonces, each device i, if its local counter $c_i > c'$, constructs the request for automatic authorisation and broadcasts its approval.

$$broadcast(i, A_i), \quad A_i = S_i(request), \quad request = (sess_{id}, nonce_1, \ldots, nonce_n) \ .$$

   If after a certain time a device did not receive all other $(n-1)$ nonces, this device aborts. Device j updates $c = c + \Delta$ when it aborts.
4. After verifying the signatures of $(k-1)$ other approvals, each device that has given its approval concludes that automatic resharing is authorised.

Fig. 1. Protocol for automatic authorisation

Note. Instead of each device placing a signature with its private key, a threshold signature protocol, for instance Shoup's protocol [13], can be deployed. Each device puts a partial signature on the request; these partial signatures are then combined to construct the signature on the request. When using a threshold signature only one signature needs to be verified. However, in the case of dishonest parties, the effort needed to verify the correctness of the threshold signature increases. For non-verifiable signature schemes, combinations of partial signatures need to verified untill a correct combination is found. For verifiable signature schemes, the partial signatures need to be verified.

3.2 Manual authorisation

Resharing can be authorised by the user, possibly altering the participating devices, e.g. adding or removing a device. The user can also be requested by GUI-enabled devices to manually authorise resharing. This will be the case if the local counter of one device reaches C or the resharing protocol failed. Adding a device is a special case and is discussed afterwards.

We distinguish three types of devices: participating, non-participating and new devices. Participating devices refer to devices that are part of the threshold secret sharing both before and after resharing. Devices no longer part of threshold secret sharing after resharing are referred to as non-participating devices; these are the devices that will be removed from the threshold secret sharing. New devices are only part of the threshold secret sharing after resharing; these are the devices that will be added to the threshold secret sharing. The proxy refers to the first device the user interacts with by entering his request for manual authorisation for resharing.

The problem of the user authenticating his request is similar to the "What You See Is What You Sign" problem [10]. The user cannot be sure that what is displayed on the screen is his request. The adversary, when controlling the device, could display another request than the one the user authenticates. The user can manifest himself towards his devices by interaction with the threshold number of these. This allows to detect cheating devices.

Since we require explicit interaction of the user before resharing takes place, we do not need to put safeguards in place to prevent DoS attacks, as is the case with automatic resharing. However, replay needs to be prevented as this could be used to annoy the user by constantly asking for interaction.

The proposed solution consists of three steps: first the user enters his request at the proxy which broadcasts the request, second he confirms his request at other devices and third the devices conclude resharing is authorised. Figure 2 shows the protocol for manual authorisation.

1. The user selects device j as the proxy and enters his request at the proxy. The proxy broadcast the user's request and approval.

$$broadcast(j, request, A_j), \quad request = (sess_{id}, \mathbf{G}), \quad \mathbf{G} = \{id_1, \ldots, id_n\},$$

$$A_j = S_j(request) \ .$$

2. Each device i verifies that it is a participating device and verifies the approval from the proxy. If correct, the request is displayed for the user to confirm.

$$user(request) \ .$$

When the user approves the request at device i, the request is signed by this device and the approval is broadcasted.

$$broadcast(i, A_i), \quad A_i = S_i(request) \ .$$

3. After verifying the signatures of $(k-1)$ other approvals, each device that has broadcasted the user's approval concludes that manual resharing is authorised.

Fig. 2. Protocol for manual authorisation

The proxy broadcasts the user's request to authorise resharing. The request consists of the identities of the participating devices. $\mathbf{G}$ is the vector that contains the identities. The request is signed by the proxy.

Each device verifies that it is a participating device and verifies the signature on the request. It is assumed that all devices in the secret sharing scheme know each other's public keys. If the signature on the request is correct, the request is displayed for the user to confirm. When a user confirms the request on a device, the request is signed by this device and this signature is broadcasted. The user confirms his request on at least $(k-1)$ participating devices; in total the user needs to interact with at least k participating devices. Before confirming, a user can verify his request at all GUI-enabled devices.

Each participating device verifies the correctness of the signatures and checks whether the request is signed by a participating device. After k valid signatures from participating devices, resharing is authorised and takes place.

Adding a device. When adding a new device, this device is not yet known to the participating devices. The user needs to verify that the device he intends to add is the device that will be added, hence this device first needs to be authenticated to the group of participating devices and vice versa. For this goal we need a way of authenticating new information, in this case public keys. We considered three possible approaches:

1. A group of at least k agreeing devices can act as a trusted entity. We only need a manual authentication protocol between the new device and this trusted entity. When performing standard *MANual Authentication* (MANA) protocols [7] between two parties we need to perform the MANA protocol at least k times, which is not user-friendly.
2. Another approach could be to only perform a MANA protocol between the new device and the proxy and verify, after the resharing protocol took place, that the new device is part of the resharing. However, this approach would introduce new weaknesses, for example, an actively compromised proxy could act as a man-in-the-middle and add an actively compromised device instead. This weakness is the result of the fact that there is no authentication between the new device and other participating devices.
3. Group manual authentication protocols, such as the ones proposed by Laur et al. [11], Nguyen et al. [12] and Wang et al. [14], allow authentication to all devices at once and do not require more effort from the user.

To exchange public keys between the new device and the participating devices, we use the *Group Message Authentication* (GMA) protocol that exchanges *Short Authenticated Strings* (SAS) by Laur and Pasini [11], shown in Fig. 3. The ordered vectors $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{r}}_i$ represent respectively all messages m_j and subkeys r_j as received by device i from devices j. The commitment in the first phase of the protocol commits a device to the final hash. Because the commitments are only opened in the second phase, no device can force the hash to a specific value.

1. R1: each participating device i broadcasts a message and a commitment value

$$broadcast(i, m_i, c_i), \quad (c_i, d_i) \leftarrow commit(i, r_i), \quad \text{random}\, r_i \ .$$

2. R2: each device i broadcasts its decommitment value and opens the commitments from each device j. **Abort if abnormal behaviour.**

$$broadcast(i, d_i), \quad (j, \hat{r}_{ji}) \leftarrow open(\hat{c}_{ji}, \hat{d}_{ji}) \ .$$

3. SAS: each device i calculates the SAS value, a SAS value is broadcasted over the authenticated channel and compared with the other SAS values.

$$user(SAS_i), \quad SAS_i = h((\mathbf{G}, \hat{\mathbf{m}}_i), \hat{\mathbf{r}}_i) \ .$$

Fig. 3. SAS-GMA proposed by Laur and Pasini [11]

The new device starts by sending a session identifier, its identity and its public key as the message in the SAS-GMA protocol with the participating devices. The message of the participating devices in the SAS-GMA protocol consists of the session identifier, the public state and their public key. The public state consists of the identities of all participating and new devices, and the threshold k. In the second round of the SAS-GMA protocol an additional check is required to detect abnormal behaviour. The new device verifies that all received messages contain the same public state. A single SAS value needs to be broadcasted over the authenticated channel: the user needs to compare the displayed values between his mobile devices.

The comparison of this SAS value is combined with the users consent for adding this device. The new device will display its SAS value, and the participating devices will ask the user if he wants to add a new device displaying the calculated SAS value. When the user confirms at device i, this device construct and signs the request and broadcasts the approval. After k valid signatures from participating devices, resharing is authorised and takes place. Figure 4 gives an overview of the protocol to manually authorise adding a device.

Other applications. The protocol for adding a device can also be used to authorise signatures or decryptions. This authorisation can be necessary to place signatures on documents or to decrypt very sensitive data. In either case the proxy has some data, unknown to the other participating devices. This unknown data is a digest of the data to be signed or an encapsulated symmetric key that needs to be decrypted to decrypt the data.

The SAS-GMA protocol provides authentic data exchange between the participating devices. The proxy's message consists of a session identifier, its identity and the data to be signed or decrypted. Other participating devices just send the session identifier as message. In round two of the SAS-GMA protocol the proxy

1. SAS-GMA R1:
   (a) new device sends the session identifier, its identity and its public key as message.
   $$m_{NEW} = sess_{id}||id_{NEW}||PK_{NEW} \ .$$
   (b) in response, each participating device i sends the session identifier, the public state and its public key as message.
   $$m_i = sess_{id}||\mathbf{G}||k||PK_i, \quad \mathbf{G} = \{id_1, \ldots, id_n, id_{NEW}\} \ .$$
2. SAS-GMA R2: **Abort if abnormal behaviour**, additional check: the new device verifies that all received messages contain the same public state.
3. SAS-GMA SAS: A single SAS message needs to be broadcasted over the authenticated channel. The user is requested by all participating devices if he wants to add the new device and if it displays the same SAS value.
   $$user(SAS_{NEW}) \ .$$
   When the user approves at participating device i, the request is constructed and signed by this device, and the approval is broadcasted.
   $$broadcast(i, A_i), \quad request = (sess_{id}, G, SAS_i), \quad A_i = S_i(request) \ .$$
4. After verifying the signatures of $(k-1)$ other approvals, each device that has broadcasted the user's approval concludes that manual resharing is authorised.

Fig. 4. Protocol for manually authorising adding a device

verifies that the session identifier matches. The request consists of the identity of the proxy, the data to be signed or decrypted, the session identifier and the SAS value.

4 Security discussion

This section discusses for the two cases of authorisation how an adversary is prevented from breaking the scheme or mounting DoS attacks.

4.1 Automatic authorisation

Since the number of devices and the threshold remain the same after resharing, the adversary can gain no advantage in the resharing protocol. The number of actively compromised devices remains the same.

Although resharing does not pose any security risk in the sense of the adversary breaking the scheme, the adversary should be prevented from being able to constantly get automatic authorisation for resharing as this would help him to set up a temporary DoS attack. For automatic authorisation at least k devices need to agree: have their local counter $c_i \geq c'$. The adversary controls $t < k$ devices and needs at least $(k - t)$ uncompromised devices with a local counter $c_i \geq c'$ to get automatic authorisation to trigger resharing. Once resharing is triggered, either it ends successfully and all uncompromised devices reset their local counter $c_i = 0$, or it is aborted and the user is alerted. Under no circumstances will the adversary succeed in getting the devices to reshare more than once.

If less than k uncompromised devices have their local counter $c_i \geq c'$, these devices can be forced to broadcast a nonce, construct the request, sign the request and broadcast the approval. Uncompromised devices having their local counter $c_i < c'$ can only be forced to send a nonce. As long as resharing is not triggered and the local counters are not updated, the adversary can force the uncompromised devices to do this indefinitely.

4.2 Manual authorisation

When the number of devices and threshold remain the same, the adversary gains no advantage by resharing. Because of the user interaction, the adversary can only mount a temporary DoS attack when sending a request and incorrect approval, which will make the other devices verify the approval but not display the request.

When allowing a user to add or remove devices while resharing we introduce a possible advantage for the adversary. When removing an uncompromised device, the total number of devices decreases while the number of actively compromised devices remains the same. When adding an actively compromised device, both the total number of devices and the total number of actively compromised devices increase. The threshold k will be set according to the number of participants.

Threshold. Threshold secret sharing schemes require that the adversary controls fewer than the threshold number of devices. To prevent the adversary from breaking the scheme, the threshold should be as high as possible. Since we also want to protect the user against permanent DoS attacks, the threshold should be as low as possible. Therefore the threshold should be the smallest majority of devices.

For an even number of devices this leads to an interesting observation: the maximum number of devices an adversary is allowed to compromise when fixing the threshold to the smallest majority is the same as when fixing the threshold to half the number of devices. In the former case this results from the condition to prevent the adversary from succeeding in a DoS permanent attack. In the latter case this follows from the condition to prevent the adversary from breaking the scheme.

Keeping in mind that usually the required effort from the adversary to succeed in a DoS attack is smaller than the effort needed to break the scheme, we opt for the latter. Furthermore this choice improves usability as authenticating a request from the user requires user interaction with the threshold number of devices.

$$k = \left\lceil \frac{n}{2} \right\rceil \qquad t_{max} = k - 1 \ . \tag{5}$$

Number of devices that can be added and/or removed. If we do not allow devices that will be removed to participate in the manual authorisation, the adversary could stop manual authentication if uncompromised devices are not part of the request. An adversary controlling $t = k - l$ devices can stop manual authorisation for resharing if the number of uncompromised devices that will be removed is greater than l. Since $t_{max} = k - 1$, at least 1 uncompromised device can be removed without the adversary having the possibility to stop it. This puts no bounds on the total number of devices that can be removed.

We do not allow the adversary to win one of his two games by resharing, conditions (1) and (2) still hold after resharing. This puts bounds on the number of devices that can be added and/or removed. For a the number of added devices, r the number of removed devices, and an adversary having actively compromised $t_a = k - l$ devices:

$$t' = t_a + a \tag{6}$$

$$n' = n + a - r \ . \tag{7}$$

$$\begin{aligned} a + r &< 2l && \text{for } n \text{ even} \\ &< 2l - 1 && \text{for } n \text{ odd} \quad . \end{aligned} \tag{8}$$

The intuition behind the difference based on the number n of devices before resharing can be explained as follows:

- for n even:
  - adding an actively compromised device: both the number of actively compromised devices and the threshold increase by one.
  - removing an uncompromised device: the number of compromised devices and the threshold remain the same.

  Conditions (1) and (2) still apply after resharing.
- for n odd:
  - adding an actively compromised device: the number of actively compromised devices increases by one while the threshold remains the same.
  - removing an uncompromised device: the number of compromised devices remains the same and the threshold decreases by one.

  Conditions (1) and (2) might not apply any longer after resharing.

Alternatively we can define a bound on the number of devices an adversary is allowed to actively compromise before resharing by fixing the number of devices a user can add and/or remove in one execution of the resharing. If we allow the user to add or remove only one device:

$$\begin{aligned} t_a < \left\lfloor \tfrac{n}{2} \right\rfloor &= k && \text{for } n \text{ even} \\ &= k - 1 && \text{for } n \text{ odd} \quad . \end{aligned} \tag{9}$$

Verifying request. Once the adversary has actively compromised the proxy, he can send an alternate request. A user will not be aware of this cheating proxy unless he verifies his request before confirming it on his devices. The adversary can get his other actively compromised devices to display the original request.

The probability of an adversary cheating successfully decreases with each device where a user verifies his request. The factor by which this probability decreases depends on the number p of participating devices, $k \leq p \leq n$. The adversary's success probability is reduced by a factor f_p, as defined in Equation (10), for each device where the user verifies his request. Because it is unknown how many devices the adversary controls, it is assumed that he controls the maximum allowed number of devices and that all these devices are participating devices.

$$f_p = \frac{p-1}{t_a - 1} \ . \tag{10}$$

In order to reduce the adversary's success probability to $0 \leq success \leq 1$, a user should verify his request at $Q_p(success)$ randomly chosen devices:

$$Q_p(success) = \lceil -log_{f_p}(success) \rceil \ . \tag{11}$$

For typical values of n and allowing only one device to be removed, the user reduces the adversary's success probability to less than 50% by verifying his request at one randomly chosen device.

Table 1 gives an overview for typical values for the number of devices. There f_n and Q_n are applicable when resharing with all devices or adding one device, f_{n-1} and Q_{n-1} are applicable when removing one device. To reduce the adversary's success probability to 10% or less, the given number of devices where a user needs to verify his request applies.

Table 1. Security overview. Let n be the number of devices, k the threshold, t the total number of compromised devices, t_a the number of actively compromised devices, f the factor by which the adversary's success probability is reduced when checking one random device and $Q(0.1)$ the number of devices needed to be checked to reduce the adversary's probability to 10%. When resharing with all devices or adding one device, the columns with subscript $_n$ are applicable; when removing one device, the columns with the subscript $_{n-1}$ are applicable.

n	k	max t	max t_a	f_n	$Q_n(.1)$	f_{n-1}	$Q_{n-1}(.1)$
3	2	1	0	-	-	-	-
4	2	1	1	∞	1	∞	1
5	3	2	1	∞	1	∞	1
6	3	2	2	5	2	4	2
7	4	3	2	6	2	5	2
8	4	3	3	3.5	2	3	3
9	5	4	3	4	2	3.5	2
10	5	4	4	3	3	2.67	3

5 Conclusion

We have presented secure and user-friendly protocols that allow authorisation for resharing in a threshold setting for Things That Think. We introduced two ways to authorise resharing: automatic and manual.

The automatic authorisation for resharing allows for periodic updates and requires no user interventions, freeing the user of any cognitive workload.

Manual authorisation provides the user with the flexibility needed to alter the set of devices that participate in the threshold secret sharing, e.g., to add or remove a device. Manual authorisation for resharing avoids confronting the user with the manual input of passwords, steering away from all related trade-offs between memorability and security.

For the manual authorisation protocol we have shown that by authorising resharing the adversary will not succeed in a permanent DoS attack or be able to break the scheme when adding or removing one device. Also the adversary will not be able to prevent manual authorisation. Towards temporary DoS attacks the adversary can only trigger parts of the protocol, which only trigger minimal energy usage at the uncompromised devices.

References

1. Internet of Things in 2020. Technical report, Joint European Commission / EPoSS Expert Workshop, 2008.
2. A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.
3. M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Third Symposium on Operating Systems Design and Implementation*, New Orleans, USA, 1999.
4. Y. Desmedt, M. Burmester, R. Safavi-Naini, and H. Wang. Threshold Things That Think (T4): Security Requirements to Cope with Theft of Handheld/Handless Internet Devices. In *Symposium on Requirements Engineering for Information Security*, West Lafayette, Indiana, USA, 2001.
5. Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical Report ISSE-TR-97-01, George Mason University, ftp://isse.gmu.edu/pub/techrep/97_01_jajodia.ps.gz, July 1997.
6. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In *IEEE Symposium on Foundations of Computer Science*, volume 1294 of *LNCS*, pages 384–393. Springer-Verlag, 1997.
7. C. Gehrmann, C. Mitchell, and K. Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, 7(1):29–37, 2004.
8. C. Gehrmann, K. Nyberg, and C. Mitchell. The personal CA–PKI for Personal Area Network. In *Proceedings of the 11th Information Society Technologies (IST) Mobile and Wireless Communications Summit*, pages 31–35, 2002.
9. B. Hardekopf, K. Kwiat, and S. Upadhyaya. A Decentralized Voting Algorithm for Increasing Dependability. In *Distributed Systems. 5th World MultiConference on Systemic, Cybernetics and Informatics (SCI2001)*, 2001.
10. P. Landrock and T. Pedersen. WYSIWYS? – What you see is what you sign? *Information Security Technical Report*, 3(2):55 – 61, 1998.

11. S. Laur and S. Pasini. SAS-Based Group Authentication and Key Agreement Protocols. In R. Cramer, editor, *11th International Workshop on Practice and Theory in Public-Key Cryptography (PKC 2008)*, volume 4939 of *Lecture Notes in Computer Science*, pages 197–213. Springer, 2008.
12. L. H. Nguyen and A. W. Roscoe. Efficient group authentication protocols based on human interaction. Cryptology ePrint Archive, Report 2009/150, 2009. `http://eprint.iacr.org/`.
13. V. Shoup. Practical Threshold Signatures. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 207–220, Bruges, Belgium, May 2000. Springer-Verlag.
14. S. Wang and R. Safavi-Naini. New Results on Unconditionally Secure Multi-receiver Manual Authentication. In *2nd International Conference on Information Theoretic Security - ICITS 2007*, LNCS. Springer-Verlag, 2007.
15. T. M. Wong, C. Wang, and J. M. Wing. Verifiable Secret Redistribution for Threshold Sharing Schemes. Technical Report CMU-CS-02-114, Carnegie Mellon University, 2002.