A Comparison-Based Methodology for the Security Assurance of Novel Systems

Peeter Laud and Jelizaveta Vakarjuk

Cybernetica AS
peeter.laud|jelizaveta.vakarjuk@cyber.ee

Abstract. In this paper, we advocate the position that the security certification of one system should make the certification of other similar systems easier, if one can present the evidence that the second system is at least as secure as the first system. We present a development of this idea, stating the components of such comparative evidence. We stretch the idea of propagating the certification to less similar systems, if one can present a sequence of systems from the certified one to the novel one, where each system is evidenced to be at least as secure as the previous one. We apply our methodology to authentication systems, where we show that a system based on threshold cryptography is at least as secure as widely used smartcard-based systems.

1 Introduction

Our critical systems are built on top of, and their security assurances depend on smaller subsystems, whose security is of utmost importance, and where the society has chosen security certification as the methodology to ensure that these subsystems satisfy the required properties. These small systems and devices include smartcards, hardware security modules, trusted execution environments, but also certain communication devices, operating systems and application software. A lot of trust is placed on the outcomes of certification, hence this process is expensive [9] and conservative [20]. When a certified system has been updated, then the new version has to pass the certification again. A system making use of novel security technologies may have hard time passing certification at all, because these technologies may not map cleanly to the categories specified in certification procedures.

In Common Criteria (CC) certification [8], we speak of *Targets of Evaluation* (ToE); this is the well-delimited system that is expected to pass certification. A *Security Target* (ST) document is compiled for certifying a particular ToE; it gives the boundaries of ToE, and lists the security requirements that ToE is expected to fulfill. These requirements are taken from standardised lists, enumerated in CC specifications. A security target may comply with, i.e contain one or more *Protection Profiles* (PP), which are themselves lists of security requirements. The security requirements are either *Security Functional Requirements* (SFR), referring to the security technologies that the ToE may use in a certain

manner in order to obtain certain security properties, or *Security Assurance Requirements* (SAR), which refer to the good practices used during the design, development, and operation of the ToE. In this paper, we use CC-related terms, but our proposed methodology should be applicable to other certification frameworks, too.

When a ToE evolves, then it may still fulfill the same SFRs listed in the ST of the old ToE, but it may fulfill them in novel manner. When a ToE makes use of novel technologies, it may be difficult to map the technical protections it offers to the fulfilment of SFRs in the ST. In both cases, it may be helpful to not match the new ToE directly against SFRs, but to compare the new ToE to the old one and make sure that the new one is *at least as secure*. This is the position we advocate in this paper: comparing two ToE-s — one with existing certification against a ST, and the other that we desire to get certified against a similar ST — can simplify the certification process, while preserving the assurances of the existing certification. The comparison establishes, that for any attack against the novel system, there exists an attack against the certified system that is no more difficult to perform, and with the same or worse consequences. We argue that such arguments should be accepted by certification bodies. Even if they are not directly accepted, the outcomes of the comparison should guide the vendor in changing the evidence documentation that supports the claims of ToE satisfying the ST.

Our proposal is similar to *game-based* security proofs of cryptographic primitives [1]. In cryptography, security definitions define an interactive attack game that is played between the adversary and the environment; the latter provides an interface to the adversary through which the primitive may be invoked in well-defined ways. The primitive is deemed secure if no adversary's probability of *winning* the game exceeds a certain value. A typical security proof presents a different game where the same adversary obviously has only a low probability of winning, and then argues that in the original game, the adversary's win is not much more likely.

In these security proofs, it is common to introduce *intermediate* games, such that any two neighbouring games differ only slightly from each other, simplifying the argument that the adversary's winning probability in i-th game is at most negligibly larger than in the $(i+1)$-st game. Here the zeroth game comes from the security definition, and the final game is the one where the adversary's win is obviously unlikely. We advocate that we can do something similar for certification: the zeroth ToE is the novel one that we want to certify, the final ToE has already been certified, and we may introduce "intermediate" ToE-s that are easier to compare against the previous and next ones.

Related work. The certification of evolving systems, the simplification of updating the evidence, and the reuse of evaluation results have been the topic of a statement from the CC Recognition Arrangement Management Committee [16]. Methods for using CC together with agile product development [18,19], and for continued certification of cyber security products [5] have been proposed. But we are aware of only a few attempts to show the security of one system by compar-

ing it to another one, where the latter one has already been deemed secure [3]. The other example is Trusted Computer System Evaluation Criteria (TCSEC), where the system evaluation process included the Rating Maintenance Phase (RAMP). The goal of RAMP was to provide an instrument to extend evaluation of a certified system to a new version of that system by analysing changes that were introduced [10].

In the rest of this paper, we describe our proposal for comparing two systems in more detail. We will then give an example of comparing two systems, where one of them is based on well-established hardware security technologies, while the other one makes use of threshold cryptography. In order to compare them against each other, we come up with a couple of intermediate systems, and argue that each of them is at least as secure as the next one.

2 Proposed methodology

Suppose we have a ToE T', for which we want to provide evidence that it matches a security target. We also have another system T, which we have already shown to match the same or a similar security target. We have the detailed specifications of both T and T', such that we can make a list of all potential weaknesses that an adversary may exploit in T or in T'. We may already have used these lists to create the security target [7]. For describing the processes of T and T', BPMN [11] may be a good choice, perhaps annotated further [14,17] to show the protection mechanisms in use. We create lists of weaknesses with high granularity, considering each data storage, movement, or processing step as something that the adversary may obtain information from, or tamper with the inputs and outputs of. The high granularity also means that an adversary may exploit one weakness either fully, or not exploit it at all, but there are no partial exploitations. Let W be the set of all weaknesses of T, and W' the set of all weaknesses of T'. Also, let $\mathcal{W}$ [resp. $\mathcal{W}'$] be the set of subsets of W [resp. W']. When an adversary performs an attack against T or T', then the set of weaknesses it exploits is an element of $\mathcal{W}$ or $\mathcal{W}'$.

The security target specifies the operational environment of the ToE, including the threats, organisational policies, and security assumptions. Hence not any set of weaknesses is considered exploitable against the system T. Rather, the security target specifies the set of *considered sets of weaknesses* $\mathcal{W}_{\mathrm{C}} \subseteq \mathcal{W}$ that contains only such sets of weaknesses that an adversary may *attempt* to exploit according to the security target. Here the attempts by the adversary do not correspond to his success; rather, the ToE is meant to employ measures that make the adversary unsuccessful. It is natural to assume that $\mathcal{W}_{\mathrm{C}}$ is downwards closed: if $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{W}$, $\mathbf{w}_1 \subseteq \mathbf{w}_2$, and $\mathbf{w}_2 \in \mathcal{W}_{\mathrm{C}}$, then also $\mathbf{w}_1 \in \mathcal{W}_{\mathrm{C}}$.

The security target for T' similarly specifies $\mathcal{W}'_{\mathrm{C}} \subseteq \mathcal{W}'$. As T' has not been certified yet, our methodology may bring adjustments to the precise definition of $\mathcal{W}'_{\mathrm{C}}$; hopefully not reducing it by too much.

To each set of weaknesses of T or T' we can attach two (abstract) quantities: the *difficulty* (for the adversary) of exploiting this set of weaknesses, and the

seriousness of the effects of successful exploitation by the adversary. We do not define the structure of these quantities, but we require that there is a (partial) order on them. Hence we get two preorders on $\mathcal{W} \cup \mathcal{W}'$: for two sets of weaknesses $\mathbf{w}_1$ and $\mathbf{w}_2$ we write $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$, if exploiting all weaknesses in $\mathbf{w}_1$ is no more difficult for the adversary than exploiting all weaknesses in $\mathbf{w}_2$. We also write $\mathbf{w}_1 \prec_\mathrm{S} \mathbf{w}_2$ if the effects of exploiting $\mathbf{w}_1$ are no worse than the effects of exploiting $\mathbf{w}_2$. A natural corollary of these definitions is, that if $\mathbf{w}_1 \subseteq \mathbf{w}_2$, then also $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$ and $\mathbf{w}_1 \prec_\mathrm{S} \mathbf{w}_2$.

It is possible that for some $\mathbf{w}_1 \neq \mathbf{w}_2$ we have both $\mathbf{w}_1 \prec_\mathrm{D} \mathbf{w}_2$ and $\mathbf{w}_2 \prec_\mathrm{D} \mathbf{w}_1$ (denote this $\mathbf{w}_1 \sim_\mathrm{D} \mathbf{w}_2$), hence $\prec_\mathrm{D}$ is not an order, but only a preorder. The same applies to $\prec_\mathrm{S}$. It may be difficult to describe all refinements that $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ have with respect to the subset inclusion. However, an imprecise description does not invalidate the outcomes of our proposed analysis.

We want to show that for each $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$, there exists some $\mathbf{w} \in \mathcal{W}_\mathrm{C}$, such that $\mathbf{w} \prec_\mathrm{D} \mathbf{w}'$ and $\mathbf{w}' \prec_\mathrm{S} \mathbf{w}$. We thus have to present a function $f : \mathcal{W}'_\mathrm{C} \to \mathcal{W}_\mathrm{C}$, such that $f(\mathbf{w}') \prec_\mathrm{D} \mathbf{w}'$ and $\mathbf{w}' \prec_\mathrm{S} f(\mathbf{w}')$ for all $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$. We have found it easier to present not a function f, but a relation $F \subseteq \mathcal{W}' \times \mathcal{W}$, requiring that each $\mathbf{w}' \in \mathcal{W}'_\mathrm{C}$ is related to at least one element of $\mathcal{W}_\mathrm{C}$.

Each element of $\mathcal{W}'$ [resp. $\mathcal{W}$] is a subset of W' [resp. W]; hence it has the natural representation as a characteristic vector of type $W' \to \{0,1\}$ [resp. $W \to \{0,1\}$]. The entire relation F is thus represented as a boolean function $\phi : ((W' \dot\cup W) \to \{0,1\}) \to \{0,1\}$, which in turn is conveniently represented as a boolean formula. The variables of this formula are exactly the weaknesses of T and T'.

We say that two systems T and T' are *similar* if there is enough relationships $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ between the corresponding attack sets. If the systems T and T' are sufficiently dissimilar, then the construction of ϕ may fail, because of the lack of relationships $\prec_\mathrm{D}$ and $\prec_\mathrm{S}$ between attack sets $\mathbf{w} \in \mathcal{W}$ and $\mathbf{w}' \in \mathcal{W}'$. Such lack of (obvious) relationships may inform us of the security mechanisms that we need to use in T' in order to follow our methodology — the security mechanisms used by T' have to be strong enough to justify the addition of $\prec_\mathrm{D}$- or $\prec_\mathrm{S}$-relationships between attack sets against T and T'. These mechanisms may also be necessary for T' to match the security target. If intermediate systems have been introduced, then the requirements to use stronger security mechanisms may propagate from T through these systems all the way back to the system T'.

3 Example

In this section, we show how to use the proposed methodology to compare two authentication systems, where the user's browser establishes a secure connection with a relying party's (RP) server, and the user is authenticated during this process. In both systems, the RP server, knowing the public key of the user, generates a challenge, which has to be signed with the corresponding private key. The systems differ in how the private key is stored, and how the signature is generated.

An authentication system has to provide processes for the full lifecycle of a user's keypair: how it is generated, how a certificate is issued for it, how the RP finds the public key, how the challenges are signed, how the key is revoked. A full comparison of the two systems has to consider all of them. For concreteness, in this paper we will only focus on the authentication process itself, which includes the signing, and also some details of the storage of the private key.

The first system T has a smartcard as its centerpiece, containing the user's private key, and issuing signatures with it when activated. There exist protection profiles for such devices [13,12], and a number of cards or chips on the card have been certified to satisfy them. A typical use of a smartcard is as part of the Client Certificate Authentication (CCA) in the Transport Layer Security (TLS) protocol [15].

The second system T' uses threshold cryptography [4], sharing the private key between the user's phone and a central server. It can be used to authenticate the client end in an established TLS session. When attempting to directly certify such a system, it may be difficult to argue that the user has sufficient control over the private key — neither the phone nor the remote server alone provides such control.

3.1 Authentication with a smartcard

When a relying party wants to make sure it is talking to a device of a user with a given public key, it will create a challenge bit-string and ask the device to sign it. If the corresponding private key is stored in a smartcard, used with a card reader, then the challenge goes from the client application to the card reader, then together with the PIN (entered by the user on the PINpad if the card reader has one, and on the keyboard otherwise) to the smartcard, which checks the PIN and creates the signature, which then moves back the same way. If there have been too many wrong entries of the PIN, then the card locks up. The whole process is displayed as a BPMN diagram in Figure 1, split into relatively atomic pieces.

The card stores the private key, and the PIN. The user also knows the PIN. Inside the card, the PIN entered by the user is compared against the PIN that the card is storing. The environment of the card is protected, meaning that the PIN and the private key (or any details about it) cannot be read out, and the logic that it executes cannot be thwarted. The card reader is also protected, in that the PIN entered on its pad will only be sent to the card and not anywhere else.

This process, and the protection offered by it is considered a good example of two-factor authentication, showing that the user knows the PIN (knowledge factor), and the user has the card (possession factor). It is considered a good example even if the PIN entry is less isolated from the outside world, i.e. when the user enters the PIN on the computer.

Weaknesses of a system with a smartcard. In this process, the following weaknesses can be identified. These are the elements of the set W — the pos-

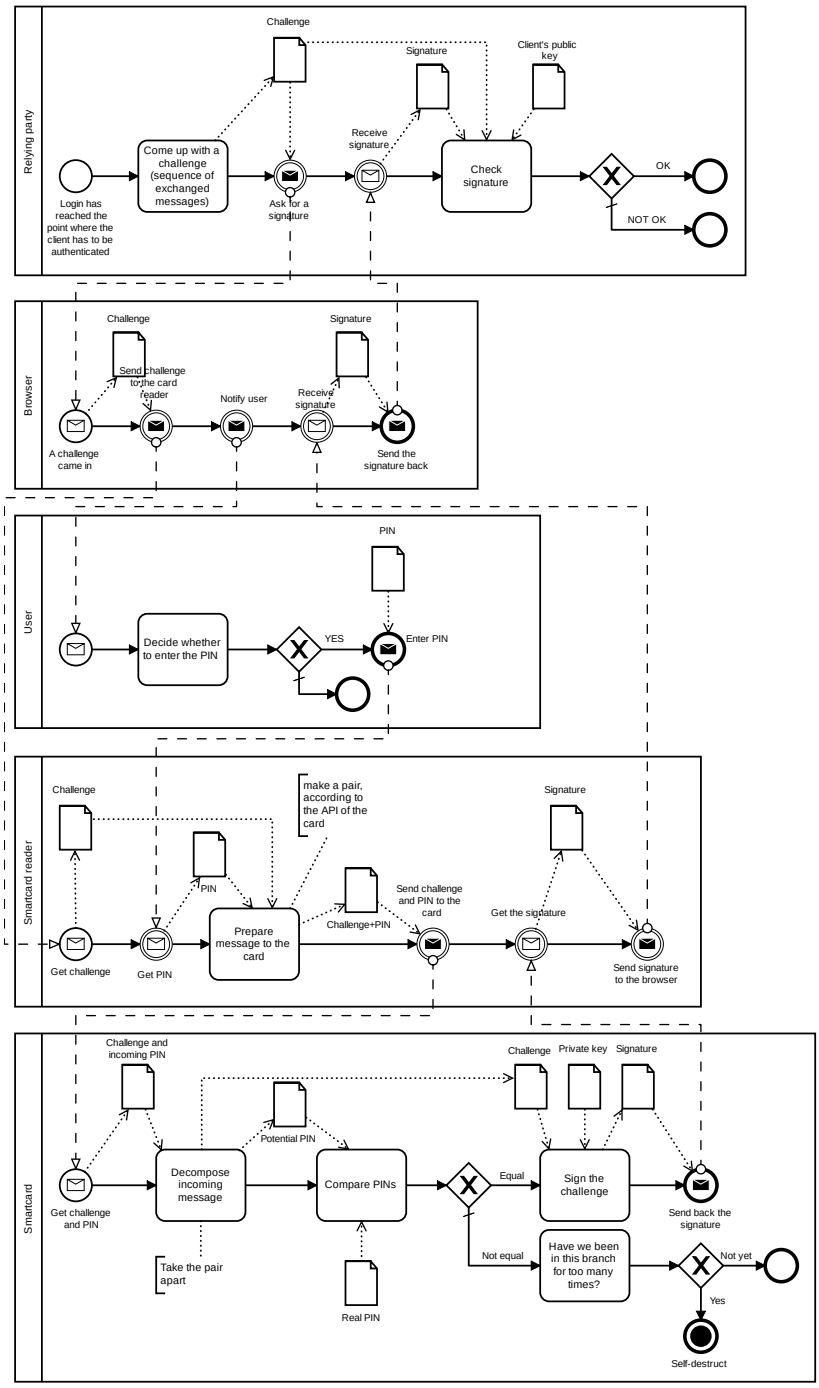

Fig. 1. The smartcard system

sible locations where an adversary could mount an attack. As the system T is considered secure, none of these locations, or their combinations are actually an exploitable weakness, i.e. $\mathcal{W}_C = \mathcal{W}$.

Possible attacks against the relying party

(RP1) Affect the computation of the challenge
(RP2) Learn the challenge
(RP3) Modify the challenge while it is sent to the browser
(RP4) Change the outcome of the signature check
(RP5) Accept the log-in, even if the signature does not check

Possible attacks against the browser

(B1) Learn the challenge
(B2) Modify the challenge while it is sent to the card reader
(B3) Modify the signature while it is sent to the relying party

Possible attacks against the user

(U1) Learn the PIN from the user
(U2) Change the PIN

Possible attacks against the smartcard reader

(SCR1) Learn the PIN that the user entered
(SCR2) Change the challenge that is sent to the smartcard
(SCR3) Change the PIN that is sent to the smartcard
(SCR4) Change the signature while it is sent to the browser

Possible attacks against the smartcard

(SC1) Learn the PIN
(SC2) Interfere with the PIN comparison procedure
(SC3) Make the decision of the PIN check take the other path
(SC4) Make the decision about counts of incorrect PINs take the other path
(SC5) Learn the private key
(SC6) Change the private key
(SC7) Change the challenge that enters the computation of the signature
(SC8) Learn the signature
(SC9) Change the signature sent back to the smartcard reader

3.2 Authentication with SplitKey

In systems relying on threshold cryptography, where the private key is stored in a secret-shared manner, several storage devices with the shares of the key have to be present in order to make use of that private key. If all the storage devices offer strong properties of unclonability, tamper-resistance and similar, then we can have even stronger security properties for the authentication process, compared to using just a single device.

However, we typically want to use threshold cryptography in order to reduce the requirements on the devices storing, and on the procedures for accessing individual shares of the private key. If we still want to argue that our authentication procedure has strong security properties similar to logging in with a smartcard, then we have to involve the properties of several devices in our arguments.

In particular, the SplitKey technology [2] for signature creation shares the private key among two devices, such that one of the devices – the phone – does not offer a strong protection for its key share. Hence, in an authentication system based on SplitKey, when arguing about the possession of a private key, we have to involve the second device in our arguments. This involvement may touch the physical properties of that device, as well as technical and organisational properties controlling the access to it.

The authentication procedure is depicted in Figure 2, in similar detail to the previous figure. Let us recall that in SplitKey, an RSA private key has been shared between the user's phone and the server, and the keyshare on the phone is encrypted with a low-entropy key (derived from the PIN that the user enters). If the adversary has obtained just the encrypted keyshare, then it cannot recognize, which PIN is the correct one. However, the server can recognize if it has received a signature share from the phone that has been created with a wrong keyshare.

When the phone has received the challenge, it asks the user to input the PIN. In order to help the user understand the context, in which the signing of the challenge is about to occur, the phone shows the user a short *control code*, derived from the challenge. The same control code is shown to the user by the relying party's website. The user is supposed to enter the PIN only if the control codes match.

Weaknesses of a SplitKey-based system. Considering the steps in Fig. 2, we can list the following possible weaknesses.

Attacks against the relying party:

(RP1) Affect the computation of the challenge
(RP2) Learn the challenge
(RP3) Affect the computation of the control code
(RP4) Learn the control code
(RP5) Modify the challenge while it is sent to the phone
(RP6) Change the outcome of the signature check
(RP7) Accept the log-in, even if the signature does not check

Attacks against the browser

(B1) Learn the control code
(B2) Change the control code, when it is shown to the user

Attacks affecting the user

(U1) Learn the PIN from the user
(U2) Change the PIN

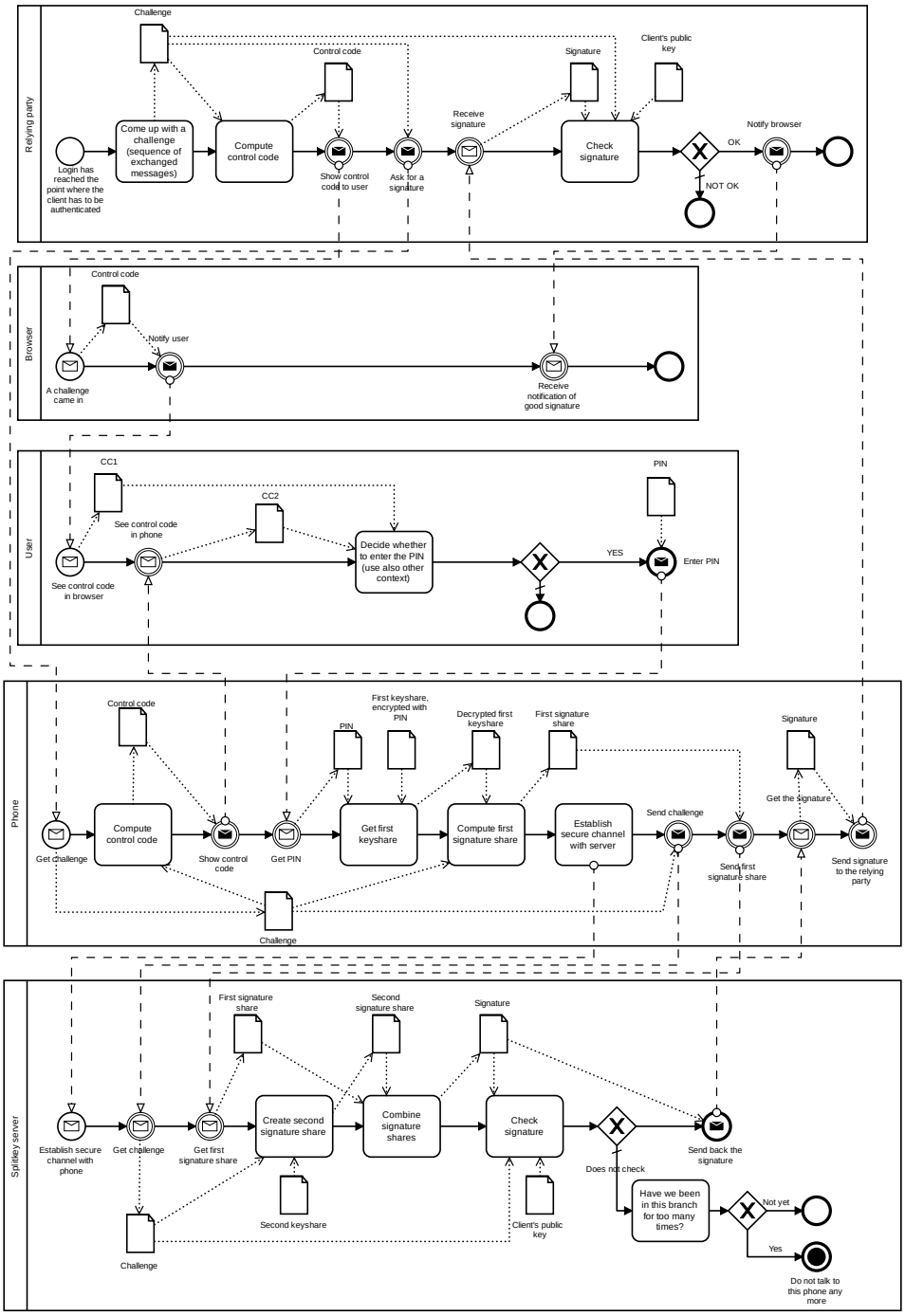

Fig. 2. SplitKey authentication system

(U3) Cause the user to incorrectly compare the control codes

Attacks against the phone

(P1) Affect the computation of the control code
(P2) Change the control code on the phone screen, where it is shown to the user
(P3) Learn the PIN that the user enters
(P4) Learn the encrypted first keyshare (encrypted with PIN)
(P5) Learn the plain first keyshare
(P6) Change the encrypted first keyshare, before it has been decrypted with the PIN
(P7) Change the plain first keyshare
(P8) Change the challenge that enters the computation of the first signature share
(P9) Learn the first signature share
(P10) Change the first signature share
(P11) Interfere with the establishment of the secure channel with the SplitKey server, thereby obtaining the capability to read and/or change messages sent over it
(P12) Change the challenge sent to the SplitKey server
(P13) Change the first signature share sent to the SplitKey server
(P14) Change the signature sent to the relying party

Attacks against the Splitkey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it
(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the second keyshare
(SS4) Change the second keyshare
(SS5) Change the challenge that goes into the second signature share creation process
(SS6) Learn the second signature share
(SS7) Interfere with the signature combination process
(SS8) Learn the signature
(SS9) Change the client's public key, against which the signature is checked
(SS10) Interfere with the signature checking procedure
(SS11) Make the decision of the signature check go otherwise
(SS12) Change the signature sent back to the phone
(SS13) Make the decision about counts of unsuccessful signature creations take the other path

Among the attacks against the SplitKey system, we can identify the following relationships for $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$. If the adversary learns both the PIN and the encrypted first keyshare, then he also has the plain first keyshare: $\{\mathrm{P5}\} \prec_{\mathrm{X}} \{\mathrm{P3}, \mathrm{P4}\}$ and $\{\mathrm{P5}\} \prec_{\mathrm{X}} \{\mathrm{U1}, \mathrm{P4}\}$, where X may be both D and S. Changing

a value has no greater effect than changing everything that is computed from this value. In particular, $\{\text{SS4}\} \prec_{\text{S}} \{\text{SS7}\} \prec_{\text{S}} \{\text{SS10}\} \prec_{\text{S}} \{\text{SS11}, \text{SS12}\}$ and $\{\text{P6}\} \prec_{\text{S}} \{\text{P7}\} \prec_{\text{S}} \{\text{P12}\} \prec_{\text{S}} \{\text{SS7}\}$. The same relationships generally continue to hold when we add the same additional attacks to both sides of $\prec_{\text{X}}$.

We are using threshold cryptography, where the adversary learning just a single share of a secret should not yet affect the seriousness of attacks that he can perform. In particular, we would like to state that $\{\text{P5}\} \cup \mathbf{w} \prec_{\text{S}} \mathbf{w}$ for a significant class of attack sets $\mathbf{w}$ that do not contain attacks against the second keyshare. We state this for all $\mathbf{w}$ that do not contain SS3 or SS6. Instead of the set $\{\text{P5}\}$, we can also consider other sets that imply the knowledge of the first keyshare or signature share: $\{\text{P3}, \text{P4}\}$ and $\{\text{P9}\}$.

3.3 First intermediate system

The systems T and T' are too different for the direct application of the comparison methodology we gave in Sec. 2. As we see below, we can cross this gap by proposing a couple of intermediate systems.

We change the system T' into the system T_1, where instead of computing the first signature share on the phone, we move all the computations into the Splitkey server. We let the Splitkey server store the user's private key share encrypted with the user's PIN and we send the PIN together with the challenge from the phone to the SplitKey server. The authentication process in the first intermediate system is depicted in Figure 3.

Weaknesses of the first intermediate system. Obviously the possible attacks against the relying party, browser, and user are the same as in the SplitKey-based system, as there have been no changes to this part of the system. Attacks against the phone and the server have changed; quite often, the change is simply in the target device.

Attacks against the phone

(P1) Affect the computation of the control code
(P2) Change the control code on the phone screen, where it is shown to the user
(P3) Learn the PIN that the user enters
(P4) Interfere with the establishment of the secure channel with the SplitKey server, thereby obtaining the capability to read and/or change messages sent over it
(P5) Change the challenge sent to the SplitKey server
(P6) Change the signature sent to the relying party
(P7) (**new**) Change the PIN sent to the SplitKey server

Attacks against the SplitKey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it

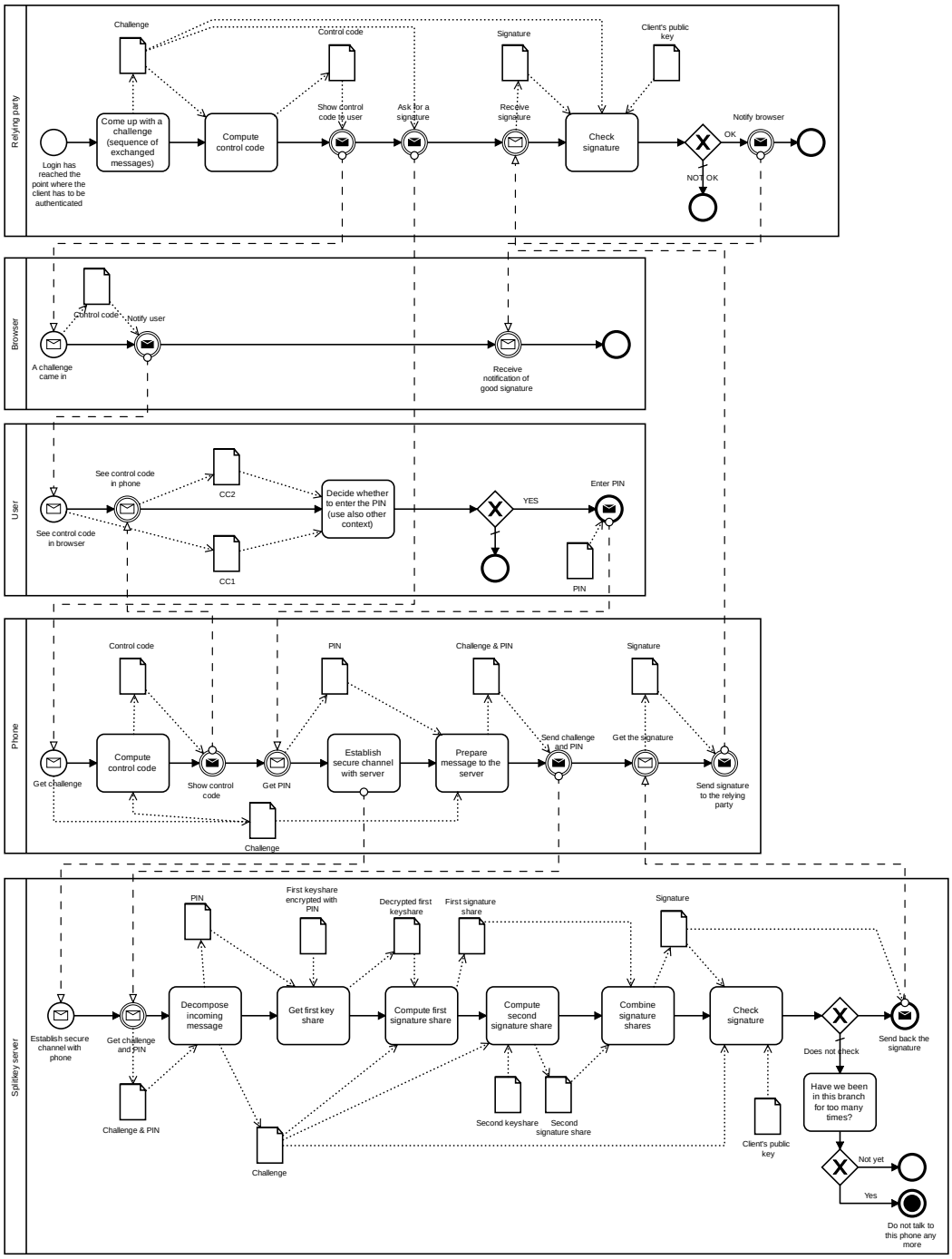

Fig. 3. The first intermediate system

(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the second keyshare
(SS4) Change the second keyshare
(SS5) Change the challenge that goes into the second signature share creation process
(SS6) Learn the second signature share
(SS7) Interfere with the signature combination process
(SS8) Learn the signature
(SS9) Change the client's public key, against which the signature is checked
(SS10) Interfere with the signature checking procedure
(SS11) Make the decision of signature check go otherwise
(SS12) Change the signature sent back to the phone
(SS13) Make the decision about counts of unsuccessful signature creations take the other path
(SS14) (**moved**) Learn the PIN received from the phone
(SS15) (**moved**) Learn the encrypted first keyshare (encrypted with PIN)
(SS16) (**moved**) Learn the plain first keyshare
(SS17) (**moved**) Change the encrypted first keyshare, before it has been decrypted with the PIN
(SS18) (**moved**) Change the plain first keyshare
(SS19) (**moved**) Change the challenge that enters the computation of the first signature share
(SS20) (**moved**) Learn the first signature share
(SS21) (**moved**) Change the first signature share

For the system T_1, we can again identify certain relationships for $\prec_{\mathrm{D}}$ and $\prec_{\mathrm{S}}$. All the relationships from T' are present, except that some of them have changed their numbers (and also moved their target from the phone to the server). We also assume that the different values the server computes during the signature creation process are equally difficult for an adversary to learn or change; thus $\{\mathrm{SS16}\} \prec_{\mathrm{D}} \{\mathrm{SS3}\}$, $\{\mathrm{SS6}\} \sim_{\mathrm{D}} \{\mathrm{SS20}\} \sim_{\mathrm{D}} \{\mathrm{SS8}\}$ and $\{\mathrm{SS5}\} \sim_{\mathrm{D}} \{\mathrm{SS19}\}$.

We also have relationships between the weaknesses of T' and T_1. We may assume that if $\mathbf{w}$ is a set of weaknesses for both of them (i.e. the names are the same), then $T'.\mathbf{w} \sim_{\mathrm{S}} T_1.\mathbf{w}$. If the targets of the weaknesses in $\mathbf{w}$ are also the same, then also $T'.\mathbf{w} \sim_{\mathrm{D}} T_1.\mathbf{w}$. For example: $\{T'.\mathrm{P4}\} \sim_{\mathrm{S}} \{T_1.\mathrm{SS15}\}$, but $\{T'.\mathrm{P4}\} \not\sim_{\mathrm{D}} \{T_1.\mathrm{SS15}\}$.

Comparison between T' and T_1. We can now write down the formula ϕ that matches each set of attacks against T' with a set of attacks against T_1. Denote $T'.\mathrm{K} \equiv (T'.\mathrm{P3} \wedge T'.\mathrm{P4}) \vee T'.\mathrm{P5}$ and $T_1.\mathrm{K} \equiv (T_1.\mathrm{SS14} \wedge T_1.\mathrm{SS15}) \vee T_1.\mathrm{SS16}$, i.e. "K" denotes the adversary's ability to learn the first keyshare. The formula ϕ is the conjunction of the following statements:

- $T'.\mathrm{RP}i \Leftrightarrow T_1.\mathrm{RP}i$ for $i \in \{1, \ldots, 7\}$
- $T'.\mathrm{B}i \Leftrightarrow T_1.\mathrm{B}i$ for $i \in \{1, 2\}$

– $T'.\mathrm{U}i \Leftrightarrow T_1.\mathrm{U}i$ for $i \in \{1, 2, 3\}$
– $T'.\mathrm{P}i \Leftrightarrow T_1.\mathrm{P}i$ for $i \in \{1, 2, 3\}$
– $T'.\mathrm{P}i \Leftrightarrow T_1.\mathrm{P}(i-7)$ for $i \in \{11, 12\}$
– $T'.\mathrm{P14} \Leftrightarrow T_1.\mathrm{P6}$
– $T'.\mathrm{SS}i \Leftrightarrow T_1.\mathrm{SS}i$ for $i \in \{1, \ldots, 13\}$
– (T'.K and T'.SS3) $\Leftrightarrow$ (T_1.K and T_1.SS3)
– (T'.K and T'.SS6) $\Leftrightarrow$ (T_1.K and T_1.SS6)
– (T'.P9 and T'.SS3) $\Leftrightarrow$ (T_1.SS20 and T_1.SS3)
– (T'.P9 and T'.SS6) $\Leftrightarrow$ (T_1.SS20 and T_1.SS6)

Sets of attacks against the relying party, browser and user have not changed, therefore the sets from the SplitKey system are equivalent to the corresponding sets in the first intermediate system. Also, certain attacks against the phone and the SplitKey server are the same in both systems as these are not connected to the changes we introduced in the first intermediate system.

Attacks involving the adversary learning one or both shares of the key and the signature are the most interesting ones. As the discussion of $\prec_{\mathrm{S}}$ above shows, we can disregard attacks where the adversary learns only one of the shares, as they are no more powerful than learning no shares at all. The last four equivalences above state that if an attack set $\mathbf{w}$ against T' allows the adversary to learn a first share (of the private key, or the signature) and a second share, then we match it with an attack set $\mathbf{w}'$ against T_1 where the adversary also learns the same first share and the same second share. The seriousness of these attacks is the same — $T'.\mathbf{w} \sim_{\mathrm{S}} T_1.\mathbf{w}'$, because the adversary obtains the same knowledge and interferes with the same operations. The difficulty is also the same, although the phone should be easier to attack than the Splitkey server — the set $\mathbf{w}$ contains either T'.SS3 or T'.SS6, hence $\mathbf{w}'$ contains either T_1.SS3 or T_1.SS6, but these are at least as difficult to perform as T_1.SS16 or T_1.SS20.

3.4 Second intermediate system

We change the system T_1 into the system T_2, where instead of computing a combined signature from two signature shares, the Splitkey server generates a single signature using a single private key. The private key is not encrypted with the user's PIN. Instead, the server verifies the correctness of the user's PIN received together with the challenge, by comparing the PIN received from the phone with one stored on the server side. The authentication process in the second intermediate system is depicted in Figure 4.

Weaknesses of the second intermediate system. Attacks against the relying party, browser, user, and phone are the same as for the first intermediate system.

Attacks against the Splitkey server

(SS1) Interfere with the establishment of the secure channel with the phone, thereby obtaining the capability to read and/or change messages sent over it

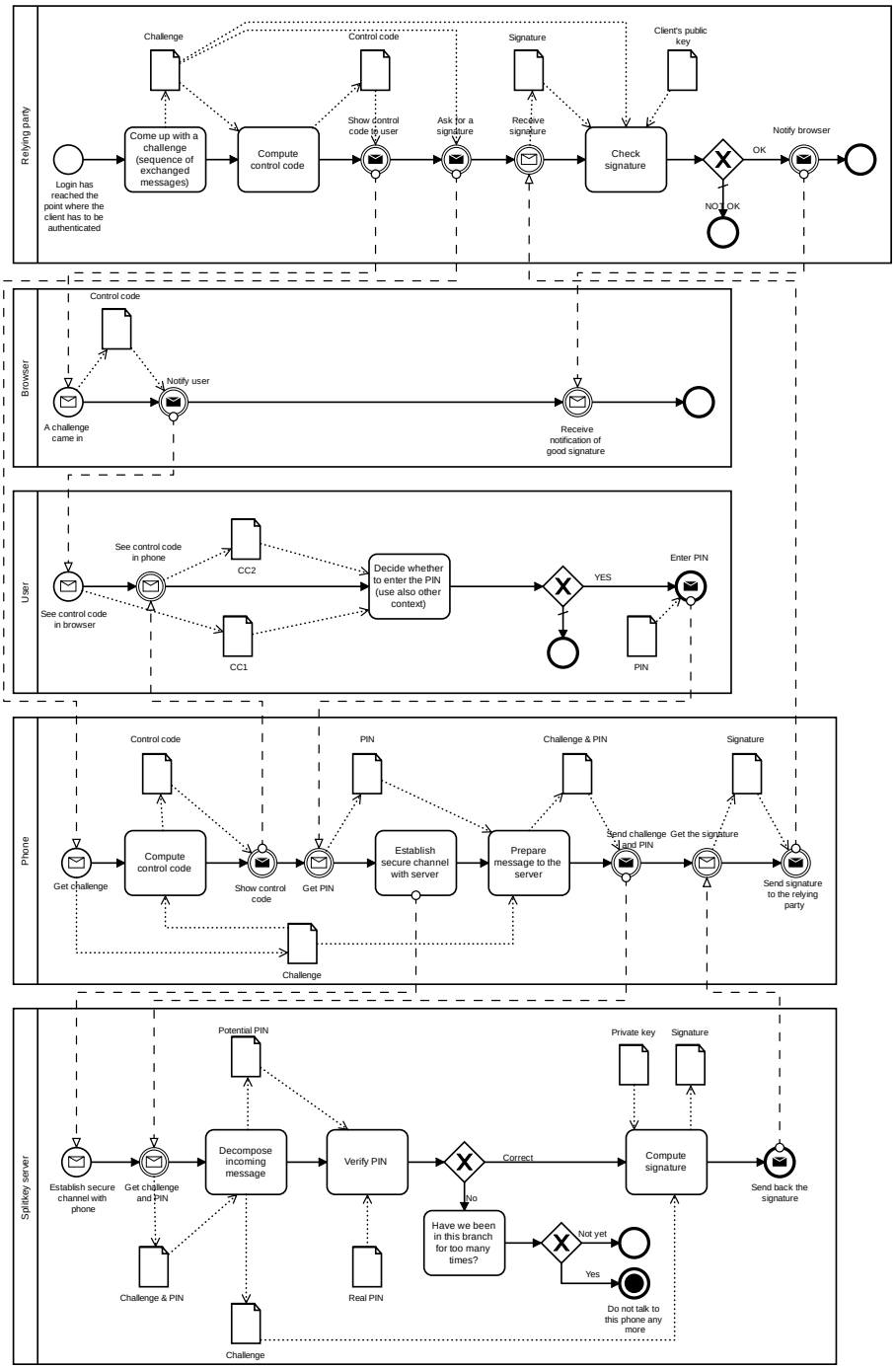

Fig. 4. The second intermediate system

(SS2) Interfere with the establishment of the secure channel with the phone, thereby confusing the server on the identity of the phone
(SS3) Learn the PIN received from the phone
(SS4) Interfere with the PIN verification process
(SS5) Make the decision of PIN check take the other path
(SS6) Make the decision about counts of incorrect PINs take the other path
(SS7) Learn private key
(SS8) Change private key
(SS9) Change the challenge that goes into the signature creation process
(SS10) Learn the signature
(SS11) Change the signature sent back to the phone

The second intermediate system is simpler than the first one because threshold cryptography is no longer used. We again have relationships between weaknesses of T_1 and T_2: we assume that learning the private key from the Splitkey server of T_2 is not harder than learning the second keyshare from the Splitkey server of T_1. We hence have $\{T_2.\text{SS7}\} \prec_\text{D} \{T_1.\text{SS3}\}$. We also have $\{T_1.\text{SS3}\} \prec_\text{S} \{T_2.\text{SS7}\}$: learning the whole private key definitely has at least as serious consequences as learning only one keyshare.

Comparison between T_1 and T_2. We can now write down the formula ϕ that matches each set of attacks against T_1 with a set of attacks against T_2. It is the conjunction of the following statements:

- $T_1.\text{RP}i \Leftrightarrow T_2.\text{RP}i$ for $i \in \{1, \ldots, 7\}$
- $T_1.\text{B}i \Leftrightarrow T_2.\text{B}i$ for $i \in \{1, 2\}$
- $T_1.\text{U}i \Leftrightarrow T_2.\text{U}i$ for $i \in \{1, 2, 3\}$
- $T_1.\text{P}i \Leftrightarrow T_2.\text{P}i$ for $i \in \{1, \ldots, 7\}$
- $T_1.\text{SS}i \Leftrightarrow T_2.\text{SS}j$ for $(i, j) \in \{(1, 1), (2, 2), (14, 3), (10, 4), (11, 5), (13, 6), (8, 10), (12, 11)\}$
- $(((T_1.\text{SS14}$ and $T_1.\text{SS15})$ or $T_1.\text{SS16}$ or $T_1.\text{SS20})$ and $(T_1.\text{SS3}$ or $T_1.\text{SS6})) \Leftrightarrow T_2.\text{SS7}$
- $(T_1.\text{SS4}$ or $T_1.\text{SS17}$ or $T_1.\text{SS18}) \Leftrightarrow T_2.\text{SS8}$
- $(T_1.\text{SS5}$ or $T_1.\text{SS19}$ or $T_1.\text{SS7}) \Leftrightarrow T_2.\text{SS9}$

Sets of attacks against the relying party, browser, user and phone have not changed as we have not introduced changes to these parts of the system. Additionally, certain attacks against the SplitKey server are the same in both systems as these are not related to the changes we introduced in T_2. In T_2, we have a single private key that is stored on the SplitKey server. It means that we can match a combination of attacks that retrieve keyshares from T_1 against a single attack in T_2, where the adversary extracts the private key from the SplitKey server. Similarly, attacks changing challenges, keyshares, or signature shares, or interfering with the combination of signatures in T_1 are no worse than the attack changing the challenge, key, or signature in T_2.

In T_1, the PIN is used to decrypt the first keyshare at the server, and only a single PIN can make the following signature check succeed. Hence it makes sense

to state that the attacks T_1.SS10 on interfering with the signature check, and T_1.SS11 on subsequently changing the control flow, are the same as the attacks T_2.SS4 and T_2.SS5 that interfere with the comparison of PINs and subsequent branching.

3.5 Comparing the second intermediate system and the smartcard system

The attacks against the smartcard system T have been described in Sec. 3.1. There is still a significant difference between T_2 and T, due to the use of different hardware components, but as we see below, we can overcome this by introducing and justifying $\prec_{\mathrm{D}}$-relationships among the attacks against T_2 and T.

Comparison between T_2 and T. The formula ϕ is the conjunction of the following statements:

- $T_2.\mathrm{RP}i \Leftrightarrow T.\mathrm{RP}i$ for $i \in \{1, 2\}$
- $T_2.\mathrm{RP}i \Leftrightarrow T.\mathrm{RP}(i-2)$ for $i \in \{5, 6, 7\}$
- $T_2.\mathrm{U}i \Leftrightarrow T.\mathrm{U}i$ for $i \in \{1, 2\}$
- $T_2.\mathrm{P}i \Leftrightarrow T.\mathrm{SCR}j$ for $(i, j) \in \{(3, 1), (5, 2), (7, 3), (6, 4)\}$
- $T_2.\mathrm{SS}i \Leftrightarrow T.\mathrm{SC}(i-2)$ for $i \in \{3, \ldots, 11\}$
- $T_2.\mathrm{P4}$ or $T_2.\mathrm{SS1} \Leftrightarrow T.\mathrm{SCR1}$ and $T.\mathrm{SCR2}$ and $T.\mathrm{SCR3}$ and $T.\mathrm{SC9}$

Most of the attacks against the relying party and user are the same in both systems. One difference comes from the fact that smartcard system does not have Control Codes, thus, the attacks against them in T_2 are vacuously successful in T. We matched a set of attacks against the phone in T_2 against similar attacks targeting smartcard reader in the smartcard system. We have not matched T_2.RP3, T_2.RP4, T_2.B1, T_2.B2, T_2.P1, and T_2.P2 with anything in T as all of these attacks are related to the Control Codes and the smartcard system does not have this protection mechanism employed.

The set of attacks against the SplitKey server in T_2 is almost identical to the set of attacks against the smartcard in the smartcard system, except attacks related to the communication channel. However, the SplitKey server and smartcard have different hardware, hence we need to introduce the relationships $\{T.\mathrm{SC}i\} \prec_{\mathrm{D}} \{T_2.\mathrm{SS}(i+2)\}$ for the analysis to go through. Also, the attacks related to the communication channel in T_2 are matched with the set of attacks where the adversary learns and changes the values sent over that channel (e.g. PIN, challenge), and this again requires extra $\prec_{\mathrm{D}}$-relationships. These requirements on $\prec_{\mathrm{D}}$ are discussed below.

3.6 Propagation of the security requirements

We described which attacks are equivalent to each other in the neighbouring systems. Now, we analyse these to identify security requirements that should be

in place in the SplitKey system for the above listed attacks to be equivalent to each other.

The communication channel between the phone and SplitKey server must offer the same level of protection as smartcard system offers against SCR1, SCR2, SCR3, SC9 attacks. This requirement arises when we move from the second intermediate system to the smartcard system, as attacks against the communication channel are removed at this point. However, we identified that attacks against the communication channel are equivalent to the attacks against the smartcard system that involve learning and modifying information sent to the smartcard. The requirement of protecting the communication channel propagates through T_2 and T_1 back to T'.

We introduced the relationship $\{T.\text{SC5}\} \prec_{\text{D}} \{T_2.\text{SS7}\}$, requiring that the Splitkey server protects the private key at least as strongly as the smartcard. The same requirement holds for T_1, except that there is no private key in T_1. We can verify that it will be sufficient to strongly protect only the second keyshare: $\{T_2.\text{SS7}\} \prec_{\text{D}} \{T_1.\text{SS3}\}$. The same requirement propagates to T'.

It should be as hard to interfere with the signature verification process in the SplitKey server as it is to interfere with the PIN verification in the smartcard. There are attacks targeting the PIN verification process in the smartcard system and in T_2. However, T_1 and T' have attacks against signature verification. Signature verification in the SplitKey system serves to verify whether the user correctly decrypted their part of the private key to create their signature share, meaning that the user entered the correct PIN code. In the smartcard system, the PIN is verified straightforwardly by comparing the code stored in the card with the one received from the user.

It should be as hard to interfere with the signature creation/combination process in the SplitKey server as it is in the smartcard system. In the smartcard system and in T_2, the adversary can modify the challenge that enters the computation of the signature. In T_1 and T', the adversary can modify challenges for two signature shares — on the phone side and on the server side. If the SplitKey server receives an incorrect signature share from the phone and decides not to discard it but use it further, it means that it can successfully execute attacks against the signature verification process, that were covered in the previous requirement. Therefore, in this requirement, we focus on the server side attacks, where the adversary can not only change the challenge for the second signature share but also interfere with the signature combination process.

It should be as hard to change the server's private key share that is used to create the signature in the Splitkey server as it is in the smartcard system. In the smartcard system and T_2, adversary can attempt to change the single private key that will be used to create signature. In T_1 and T', the adversary may target two shares of the private key. With the modified user's share of the private key, adversary will create a signature share that will not pass the verification procedure. If the server decides to use incorrect signature share, it means that one can execute attacks against signature verification that were covered before. Therefore, this requirement again focuses on the modifications on the server side.

It should be as hard to capture the PIN entered by the user from the phone as it is to capture the PIN from the smartcard reader. Starting from the smartcard system, where the PIN is sent to the smartcard through the smartcard reader this requirement propagates all the way back to the SplitKey system, where the user enters the PIN in the phone. It may be difficult to argue that this requirement is satisfied, if a smartcard reader with PINpad is used in T. It will be easier to argue for it, if computer's keyboard is used in T.

4 Conclusion

We have presented the case that comparison-based arguments can be useful for providing the evidence that a system satisfies certain security properties, and, in particular, that a ToE satisfies a Security Target. It remains to be determined, how scalable the proposed methodology is – how much effort this methodology will require in different practical applications for evolving and novel ToEs, and what kind of tool support will be useful. The proposed methodology also needs evaluation by certification laboratories.

In our example, we have compared authentication systems. We hope that it is possible to similarly show that a threshold cryptography based system may be a qualified electronic signature creation device (QSCD) [6], by again comparing it to systems making use of smartcards.

References

1. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
2. Buldas, A., Kalu, A., Laud, P., Oruaas, M.: Server-Supported RSA Signatures for Mobile Devices. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10492, pp. 315–333. Springer (2017)
3. Buldas, A., Saarepera, M.: Electronic Signature System with Small Number of Private Keys. In: Ellison, C.M., Polk, W.T., Hastings, N.E., Smith, S.W. (eds.) NISTIR 7085: 2nd Annual PKI Research Workshop Proceedings, pp. 110–122. National Institute of Standards and Technology (NIST) (2004)
4. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC '94). p. 522–533. Association for Computing Machinery, New York, NY, USA (1994)
5. Dupont, S., Ginis, G., Malacario, M., Porretti, C., Maunero, N., Ponsard, C., Massonet, P.: Incremental Common Criteria Certification Processes using DevSecOps Practices. In: IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021. pp. 12–23. IEEE (2021)

6. European Parliament and Council of European Union: Regulation (EU) no 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. OJ L 257, 28.8.2014, p. 73-114 (2014)
7. Hernandez-Ardieta, J.L., Blanco, P., Vara, D.: A methodology to construct Common Criteria security targets through formal risk analysis. In: Proceedings of XII Spanish Meeting on Cryptology and Information Security (RECSI 2012) (2012)
8. ISO/IEC 15408-1/2/3:2005 - Information technology — Security techniques — Evaluation criteria for IT security
9. Keblawi, F., Sullivan, D.: Applying the Common Criteria in Systems Engineering. IEEE Security and Privacy **4**(2), 50–55 (2006)
10. National Computer Security Center: Rating Maintenance Phase Program Document Version 2. Rainbow Series, NCSC-TG-013 V2 (1995), https://web.archive.org/web/20110720184904/http://iaarchive.fi/Rainbow/NCSC-TG-013%20PINK%20version%202.pdf
11. OMG: Business Process Model and Notation (BPMN), http://www.omg.org/spec/BPMN/2.0/
12. PP-Module for User Authentication Devices, Version 1.0. National Information Assurance Partnership (Jul 2019)
13. prEN 14169-1:2009: Protection profiles for Secure signature creation device — Part 2: Device with key generation. Technical Committee CEN/TC 224 (Dec 2009)
14. Pullonen, P., Matulevičius, R., Bogdanov, D.: PE-BPMN: privacy-enhanced business process model and notation. In: Carmona, J., Engels, G., Kumar, A. (eds.) Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10445, pp. 40–56. Springer (2017)
15. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446, https://www.rfc-editor.org/info/rfc8446
16. Reuse of Evaluation Results and Evidence (Oct 26th 2002), information Statement on behalf of the Common Criteria Recognition Arrangement Management Committee, Document no. 2002-08-009-002
17. Salnitri, M., Dalpiaz, F., Giorgini, P.: Designing secure business processes with SecBPMN. Softw. Syst. Model. **16**(3), 737–757 (2017)
18. Sinnhofer, A.D., Raschke, W., Steger, C., Kreiner, C.: Evaluation paradigm selection according to Common Criteria for an incremental product development. In: Tverdyshev, S. (ed.) International Workshop on MILS: Architecture and Assurance for Secure Systems, MILS@HiPEAC 2015, Amsterdam, The Netherlands, January 20, 2015. Zenodo (2015)
19. Sinnhofer, A.D., Raschke, W., Steger, C., Kreiner, C.: Patterns for Common Criteria Certification. In: Link, C., Eloranta, V. (eds.) Proceedings of the 20th European Conference on Pattern Languages of Programs, EuroPLoP 2015, Kaufbeuren, Germany, July 8-12, 2015. pp. 33:1–33:15. ACM (2015)
20. Sun, N., Li, C., Chan, H., Le, B.D., Islam, M.Z., Zhang, L.Y., Islam, M.R., Armstrong, W.: Defining security requirements with the common criteria: Applications, adoptions, and challenges. IEEE Access **10**, 44756–44777 (2022)