Secrecy by typing in the computational model : the signature case

Zacharie Moughanim
ENS Rennes
Rennes, France

Supervised by Stéphanie Delaune, Clément Hérouard and Joseph Lallemand

***Abstract*—We introduce an extension to a technique developed in the CCSA logic to automatically prove a weak notion of secrecy for signature protocols, extending the already existing technique by Delaune, Hérouard and Lallemand. The original work considered symmetric and asymmetric encryption only, we here aim to extend it with a signature support.**

I. Introduction

In the past decades, the multiplicity of communication and therefore of communication protocols stressed the importance of secure cryptographic protocols, sheding light on formal proofs of such protocols.

The Computationally Complete Symbolic Attacker (CCSA) framework [BC14] is a new approach to protocol verification, which was implemented in the SQUIRREL proof assistant [Bae+21]. This tool aims to provide a framework to prove various properties on cryptographic protocols, in particular, *indistinguishability*. This property is used to express other security guarantees, for instance *strong secrecy*, namely when a message is indistinguishable from a perfectly random message. Complementarily, there is another property, *weak secrecy* based upon the notion of non-deducibility. A message is *weak secret* when it cannot be fully computed by an attacker. Even though this last property is weaker, it is often the first step towards a strong secrecy proof. The goal here is to prove weak secrecy of certain messages in protocols involving signatures.

Example 1: We consider a simplified version of the Denning-Sacco key exchange protocol [Bla08, DS81], which we informally describe below:

$$A \to B : \{\!|\mathsf{sign}(\langle A, B, N\rangle, \mathrm{ssk})|\!\}_{\mathrm{pkB}}$$

In this protocol, we would like to show the exchanged nonce N is secret. Intuitively, secrecy is preserved because, since the message was signed, we know A, B, N couldn't have been tampered with by the attacker, and so this message was indeed sent by A, and destined to B.

SQUIRREL does not provide as extensive a support for non-deducibility as for indistinguishability. An automatic way to prove weak secrecy in the proof assistant SQUIRREL was introduced [DHL25], and the authors claimed it would not be to be excessively difficult to add another primitive to their work. By adding support for signature primitives, we thus provide a first positive answer to this claim in addition to the results themselves.

II. Overview of previous result

A messages is represented by a term of the CCSA logic t. Such a term is interpreted as a Probabilistic Polynomial Turing Machines (PPTM) $[\![t]\!]^{\mathbb{M}}(1^\eta, \rho)$.

We write $t \blacktriangleright u$ whenever u is non-deducible from t, i.e. when for any PPTM attacker $\mathcal{A}$, the probability that $\mathcal{A}([\![t]\!]^{\mathbb{M}}(1^\eta, \rho)) = [\![u]\!]^{\mathbb{M}}(1^\eta, \rho)$ is negligible. We write $t \approx u$, and say those terms are equivalent, whenever the probability that they are equal is overwhelming.

The previous work [DHL25] proves weak secrecy relying on *typing*.

We set several *message types*, for instance:

- Low: the type of public messages; every message sent in the network is required to be of this type;
- High: the type of secret messages;
- Bool: boolean messages.

We also have several key types to type specially keys used in cryptographic primitives: SK[T], AK[T].

A typing sequent is of the following form: $\Gamma; R \vdash t : \mathrm{T}$, with t a term, T a message type, Γ mapping variables, names and keys to types, and R the set of random used in cryptographic primitives.

The main result derives from the following theorem, which states that a message typed High cannot be deduced by an attacker from a message typed Low.

Theorem 1: For all terms t_L, t_H such that $\Gamma; R \vdash t : \mathrm{Low}$ and $\Gamma; R \vdash t : \mathrm{High}$, we have $t_L \blacktriangleright t_H$.

The proof is modular; it is divided in three distinct parts:

- Firstly, we prove the result for a type system restricted to encryption function (we forbid decrypting messages).
- Then, we reduce the system of all messages to the restricted one, by expressing functions we omitted with functions of the restricted system.
- Finally, we extend the result to protocols, and not only messages.

This last part is actually generic enough we don't have to modify it to preserve the result for signature as well,

once the first two parts are modified to take signature into account.

III. Liminary: properties of signature

We refer the reader to the original paper [DHL25], and we keep the same notations and definitions. We also keep lemma numbering consistent with this paper.

Before adding signature primitives to the type system; we add three functions: $\mathsf{sign}/2$ the signature function, $\mathsf{ver}/3$, to check a signature against a message, and $\mathsf{vk}/1$, producing the verification key of a given signature key.

For clarity purpose, we give here the properties we assume upon the signature scheme.

- **Correction.** We assume ver and sign satisfy the equation:
$$[\![\mathsf{ver}(\mathsf{sign}(m,k),m,\mathsf{vk}(k))]\!]^{\mathbb{M}}(1^\eta,\rho) = [\![\mathsf{true}]\!]^{\mathbb{M}} \quad (1)$$
- **Verification type.** As expected, ver returns a boolean:
$$[\![\mathsf{ver}(s,m,k)]\!]^{\mathbb{M}}(1^\eta,\rho) \in \{[\![\mathsf{false}]\!]^{\mathbb{M}}, [\![\mathsf{true}]\!]^{\mathbb{M}}\} \quad (2)$$
- **Cryptographic assumption.** The signature scheme satisfies the Euf-Cma (Existential Unforgeability Agains Chosen Message Attack) game, an usual assumption on signature ensuring an attacker without the signature key cannot forge a valid signature. A formal definition is recalled in Appendix A (see Game 1).

IV. The additional rules

First, we add a new key type for signature: SSK[T]. The additional rules are presented in Fig. 1 and Fig. 2.

$$\frac{\Gamma(\mathsf{k}) = \mathrm{SSK[T]}}{\Gamma;\mathrm{R} \vdash \mathsf{vk}(\mathsf{k}[\vec{i}]) : \mathrm{Low}}\ \textsc{Vk}$$

$$\frac{\Gamma;R \vdash t : \mathrm{T} \qquad \Gamma(\mathsf{k}) = \mathrm{SSK[T]}}{\Gamma;\mathrm{R} \vdash \mathsf{sign}(t,\mathsf{k}[\vec{i}]) : \mathrm{Msg}}\ \textsc{Sign-Msg}$$

$$\frac{\Gamma;R \vdash t : \mathrm{T}' \qquad \Gamma(\mathsf{k}) = \mathrm{SSK[T]} \qquad \mathrm{T}' \leq \mathrm{Low} \qquad \mathrm{T}' \leq \mathrm{T}}{\Gamma;\mathrm{R} \vdash \mathsf{sign}(t,\mathsf{k}[\vec{i}]) : \mathrm{Low}}\ \textsc{Sign-Low}$$

$$\frac{\Gamma;R_1 \vdash t : \mathrm{Msg} \qquad \Gamma;R_2 \vdash u : \mathrm{Msg} \qquad \Gamma;R_3 \vdash vk : \mathrm{Msg}}{\Gamma;R_1 \sqcup R_2 \sqcup R_3 \vdash \mathsf{ver}(t,u,vk) : \mathrm{Bool}}\ \textsc{Ver-Bool}$$

Fig. 1: Signature rules

We have two rule to type signatures: Sign-Msg and Sign-Low. We need rule Sign-Msg to be able to sign any messages: otherwise, we would only be able to sign public messages. However, if we only had rule Sign-Msg, an action whose output is simply the signature of a message could not be typed, since it would have type $\mathrm{Msg} \neq \mathrm{Low}$. Therefore, we additionally need Sign-Low to be able to correctly type protocols.

$$\frac{\Gamma, x : \mathrm{U};R_1 \vdash s : \mathrm{Msg} \qquad \Gamma, x : \mathrm{U}';R_2 \vdash C[\mathsf{true}] : \mathrm{T} \qquad \Gamma, x : \mathrm{U};R_3 \vdash C[\mathsf{false}] : \mathrm{T} \qquad \Gamma(\mathsf{k}) = \mathrm{SSK[U']}}{\Gamma, x : \mathrm{U};R_1 \sqcup R_2 \sqcup R_3 \vdash C[\mathsf{ver}(s,x,\mathsf{vk}(\mathsf{k}[\vec{i}]))] : \mathrm{T}}\ \textsc{Ver}$$

Fig. 2: Verification rule for secrecy

This rule is rather peculiar: it is the only one that refers to more than just the top-level subterms. As in rule Adec and Sdec, checking a signature make us learn some information about what we check.

However, since the useful part of the verification is not only the output of ver, but the message we verify, we learn a type information about x, and not about $\mathsf{ver}(s,m,\mathsf{vk}(k))$.

Example 2: Returning to the Denning-Sacco protocol. We require that N is secret, thus, we set $\Gamma(N) = \mathrm{High}$. We focus on the typing of the signature: $\mathsf{sign}(\langle A,B,N\rangle, \mathrm{ssk})$. Since $\langle A,B,N\rangle$ contains N of type High, we cannot use Sign-Low, because we will sign a message of type $\mathrm{Cst}^\infty_\mathrm{A} \times \mathrm{Cst}^\infty_\mathrm{B} \times \mathrm{High} \not\leq \mathrm{Low}$. Finally, by setting $\Gamma(\mathrm{ssk}) = \mathrm{SSK}[\mathrm{Cst}^\infty_\mathrm{A} \times \mathrm{Cst}^\infty_\mathrm{B} \times \mathrm{High}]$, we can type $\mathsf{sign}(\langle A,B,N\rangle, \mathrm{ssk})$ of type Msg with rule Sign-Msg.

V. Restricted type system

In this first section, we work with random mappings rather than sets of randoms. A function $\mathcal{R}$ mapping a random to the pair (m,k) means we authorize to encrypt (m,k) with said random. The additional rules of the restricted system are those in Fig. 1 where we replaced random sets by random mappings.

Notice all rules presented manipulate meta-terms (with indices as parameters), but since we do not consider the extension of the results to protocols here, we will work only with base-logic terms throughout this paper (basically, terms where we remove indices).

We begin by checking lemmas of the restricted system. The new rules do not add much more work, since the signature rules of the restricted system does not contain much cryptographic knowledge, plus, sign does not use randoms.

We denote by $\Gamma;\mathcal{R} \vdash_\mathrm{r} t : \mathrm{T}$ a typing sequent of this restricted system. We introduce a semantic counterpart, $\Gamma;\mathcal{R} \vDash t : \mathrm{T}$, meaning t has the expected behavior for a term of type T e.g. if $\mathrm{T} = \mathrm{High}$, t cannot be deduced by an attacker.

The main goal of this section is to prove the following theorem:

Theorem 2: Let $(\Gamma;\mathcal{R})$ be a well-formed mapping environment, t a ground base logic term, and T a message type. If $\Gamma;\mathcal{R} \vdash_\mathrm{r} t : \mathrm{T}$ then $\Gamma;\mathcal{R} \vDash t : \mathrm{T}$.

We will only write lemmas where an actual change take place, and give a proof sketch for each of them.

The following lemma states each secret message subterm of a public message is actually "hidden" behind an operation that can only leak a negligible number of bits on the secret e.g. an equality or an encryption.

Lemma 6.: Let $(\Gamma;\mathcal{R})$ be a well-formed mapping environment, and t_L be a ground term such that $\Gamma;\mathcal{R} \vdash_{\mathrm{r}} t_L : \mathsf{Low}$. If $t_L = C[t_H]$, for a t_H such that $\Gamma;\mathcal{R} \vdash_{\mathrm{r}} t_H : \mathsf{High}$ and a context C, then C has one of the following forms:

- $C_1[\mathsf{senc}(C_2, k, r)]$, for some contexts C_1, C_2 with $\Gamma(k) = \mathsf{SK[T]}$ for some T;
- $C_1[\mathsf{zeros}(C_2)]$ for some C_1, C_2;
- $C_1[C_2 = t]$ for some C_1, C_2, t;
- $C_1[t = C_2]$ for some C_1, C_2, t;
- $C_1[\mathsf{ver}(C_2, u, \mathsf{vk}(k))]$ for some C_1, C_2, u;
- $C_1[\mathsf{ver}(t, C_2, \mathsf{vk}(k))]$ for some C_1, C_2, t;

Proof sketch: The only new rule that can produce a term of type Low from subterms that are not already Low is rule Ver-Bool, since $\mathsf{Bool} \leq \mathsf{Low}$, this is handled by the two additional case we added in the lemma. □

The next definition describes an algorithm that can be used by the attacker given some oracles, to compute the interpretation of certain terms. This will be the key to our reductions to cryptographic games later.

Definition 1. (Evaluator with signatures): Let $\mathcal{O} = \{\mathcal{O}_{\mathsf{name}}, \mathcal{O}_{\mathsf{senc}}, \mathcal{O}_{\mathsf{aenc}}, \mathcal{O}_{\mathsf{adec}}, \mathcal{O}_{\mathsf{pk}}, \mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{vk}}\}$. The evaluator $\mathcal{E}^{\mathcal{O}}_{\mathbb{M},\Gamma,\mathcal{R}}(t)$ on input t is a recursive algorithm on t:

- Cases of adec, senc, etc. [DHL25]
- $\mathcal{E}^{\mathcal{O}}_{\mathbb{M},\Gamma,\mathcal{R}}(\mathsf{vk}(k))(1^\eta, \rho_a)$ returns the result of $\mathcal{O}_{\mathsf{vk}}(\texttt{"}k\texttt{"})$.
- If $\Gamma(k) = \mathsf{SSK[T]}$, then $\mathcal{E}^{\mathcal{O}}_{\mathbb{M},\Gamma,\mathcal{R}}(\mathsf{sign}(m,k))(1^\eta, \rho_a)$ first computes $p = \mathcal{E}^{\mathcal{O}}_{\mathbb{M},\Gamma,\mathcal{R}}(m)$ and then returns the result of $\mathcal{O}_{\mathsf{sign}}(p, \texttt{"}k\texttt{"})$.

The new oracles behave as follows:

- $\mathcal{O}_{\mathsf{sign}}(p, \texttt{"}k\texttt{"})$ returns $[\![\mathsf{sign}]\!]^{\mathbb{M}}(1^\eta, \rho)(p, [\![k]\!]^{\mathbb{M}}(1^\eta, \rho))$
- $\mathcal{O}_{\mathsf{vk}}(\texttt{"}k\texttt{"})$ returns $[\![\mathsf{vk}(k)]\!]^{\mathbb{M}}(1^\eta, \rho)$ if $\Gamma(k) = \mathsf{SSK[T]}$ for some T and fail otherwise.

We say a term t is evaluable when the probability that the evaluator returns the expected result $[\![t]\!]^{\mathbb{M}}(1^\eta, \rho)$ is overwhelming.

Lemma 7.: Let $(\Gamma;\mathcal{R})$ be a well-formed mapping environment, t a ground base logic term such that $\Gamma;\mathcal{R} \vdash_{\mathrm{r}} t : \mathsf{T}$ for some message type T. We have that t is evaluable in $(\Gamma;\mathcal{R})$.

Proof sketch: The additional cases are straightforward new induction case. □

Now we modified and proved the lemmas of the restricted system, we can put everything back together and show the main theorem.

Proof of Theorem 2: We detail the four new cases, and the case Name for which there is a slight change:

- Vk, Sign and Sign-Low: Since the resulting type is Msg or Low, we have nothing more to prove.
- Ver-Bool: We ought to prove that $\mathbb{P}([\![\mathsf{ver}(t,u,\mathsf{vk})]\!]^{\mathbb{M}}(1^\eta,\rho) = [\![\mathsf{false}]\!]^{\mathbb{M}}(1^\eta,\rho) \vee [\![\mathsf{ver}(t,u,\mathsf{vk})]\!]^{\mathbb{M}}(1^\eta,\rho) = [\![\mathsf{true}]\!]^{\mathbb{M}}(1^\eta,\rho)) \in \mathsf{ow}(\eta)$, which is an immediate consequence of (2).
- Name: Given a term t_L typed Low, we have to show $t_L \blacktriangleright n$ if $\Gamma(n) = \mathsf{High}$. By Lemma 6., t_L contains n only under $=, \mathsf{zeros}, \mathsf{senc}$ or ver. We can actually replace t_L by another term $\tilde{t_L}$ containing n only under $=, \mathsf{zeros}$ or ver. We construct $\mathcal{B}$, who's trying to compute $\tilde{t_L}$. When encountering a verification $\mathsf{ver}(t, t', \mathsf{vk})$ with t or t' containing n, $\mathcal{B}$ draws a random boolean and return it, as in the case of an equality. The remaining of the proof follows.

□

Therefore, adding rules Sign-Msg, Sign-Low, Ver-Bool and Vk preserves soundness of the type system; notice however we didn't add any cryptographic reductions or any substantial new elements to proofs, since these rules does not exhibit any cryptographic content. The purpose of the following subsection is to prove a typing rule containing the knowledge given by the cryptographic assumption.

VI. Base-level type system

In this section, we consider the base-level type system [DHL25], to which we add rule Ver Fig. 2 and the additional rules from the restricted system Fig. 1. We denote $\Gamma;R \vdash_{\mathrm{b}} t : \mathsf{T}$ a typing sequent in this fragment. The main goal of this section is to establish the following theorem.

Theorem 3: Let $(\Gamma;R)$ be a well-formed typing environment, t a ground base logic term, and T a message type. If $\Gamma;R \vdash_{\mathrm{b}} t : \mathsf{T}$, then there is a $\mathcal{R}$ such that $\Gamma;\mathcal{R} \vDash t : \mathsf{T}$.

This is basically the same as Theorem 2. We want the rules we add in the base-level system to preserve correctness.

This section contains the most intricate proofs. Similarly as in the case of Sdec, where we use the Int-Ctxt assumption to prove the correctness, we here have to prove a similar reduction to the Euf-Cma game.

The main point is the proof of correctness of rule Ver. This rule contains all the knowledge provided by the

signature. First, for a verification, and more generally, for any boolean, one may first consider the case where the result is true, and then consider the case where the result is false. Now, when encoutering a signature verification, assuming it returns true, we *gain knowledge* that the message we check it against is of the type the signature key excepts; in other words, the signature check ensures that the attacker did not modify the type of the message.

A. Cryptographic reduction

As for encryption, we begin by defining an idealized version of the primitives. Here, given a verification $\mathsf{ver}(s, m, k)$, we want to get the term that was signed to get s and whose interpretation is equal to m, if it exists.

Definition 2.: Let s, m be ground base logic terms. Let $\{\mathsf{sign}(m_i, k) \mid i \in \{1, ..., n\}\}$ be all subterms of m and s of this form, assuming this set is nonempty, we set :

$$\mu(s, m, k) \stackrel{\text{def}}{=} \begin{cases} \text{if } m{=}m_1 \text{ then } m_1 \text{ else} \\ \quad \text{if } m{=}m_2 \text{ then } m_2 \text{ else} \\ \quad\quad ... \\ \quad\quad\quad \text{if } m{=}m_n \text{ then } m_n \text{ else} \\ \quad\quad\quad\quad m_n \end{cases}$$

This term preserves good properties: it must be equal to a term that was signed, thus it has the type of a signed message. From a protocol point of view, this reflects the fact that a message whose signature is checked must be a message that was sent by the owner of the signature key, which corresponds to the property we intuitively require of a signature scheme.

We now proceed to show our main result about signature: when the verification succeeds, the message we verify is equivalent to $\mu(s, m, k)$.

Lemma A.: Let m be a ground base logic term, C a context and s a ground base logic term such that $\mathrm{fv}(C) = \{x\}$, T a type and $(\Gamma, \mathcal{R})$ a well-formed mapping environment such that $\Gamma(k) = \mathrm{SSK[T]}$. If $\Gamma; \mathcal{R} \vdash_r s : \mathrm{Msg}$ and $\Gamma; \mathcal{R} \vdash_r m : \mathrm{Msg}$, then

- If m and s have no subterm of the form $\mathsf{sign}(t, k)$, then $C[\mathsf{ver}(s, x, \mathsf{vk}(k))]\{x \mapsto m\} \approx C[\mathsf{false}]\{x \mapsto m\}$
- Otherwise,

$$C[\mathsf{ver}(s, x, \mathsf{vk}(k))]\{x \mapsto m\} \approx \begin{cases} \mathsf{if}\ \mathsf{ver}(s, m, \mathsf{vk}(k))\ \mathsf{then} \\ \quad C[\mathsf{true}]\{x \mapsto \mu(s, m, k)\} \\ \mathsf{else} \\ \quad C[\mathsf{false}]\{x \mapsto m\} \end{cases}$$

Proof: We only detail the second case, the first one is easier and the proof technique is similar.

The main argument is that under the hypothesis $[\![\mathsf{ver}(s, m, \mathsf{vk}(k))]\!]^{\mathbb{M}}(1^\eta, \rho) = [\![\mathsf{true}]\!]^{\mathbb{M}}$, we have $m \approx \mu(s, m, k)$. We prove this by constructing an attacker agains the Euf-Cma game. This attacker will simply try to compute the pair (m, s) simulating the evaluator, using the oracle of the game to answer queries to the signature oracle and submit it to the game. We show the advantage of this attacker is complementary to the probability we want to show is overwhelming. □

B. Main result

Now we have established this result for ver, we can prove the main result of this section.

We first show that every term well-typed in the base-level type system can be transformed into another term, typed in the restricted system with the same type. Since the restricted system only deals with ground term, we introduce the notion of well-typed substitution.

A *substitution* θ is a mapping of variables to ground base logic terms, $\mathrm{dom}(\theta)$ denotes its domain, $t\theta$ its application to a term. It is *grounding* for a term whenever its application to this term produce a ground term, and it is well-typed when every term in its image is.

To be able to carry out the induction when encountering rule Assign, we generalize the induction hypothesis to any term together with a well-typed substitution.

Lemma 17.: Let t be a term of the base logic, (Γ, R) be a well-formed typing environment, $(\Gamma, \mathcal{R})$ be a mapping typing environment that is also well-formed such that $\mathrm{dom}(\mathcal{R}) \cap R = \emptyset$, and θ be a substitution grounding for t. If $\Gamma; R \vdash_b t : \mathrm{T}$ and $\Gamma; \mathcal{R} \vdash_r \theta$, then there exist a ground term t', and a mapping $\mathcal{R}'$ such that:

- $\mathrm{dom}(\mathcal{R}') \subseteq R$;
- $t\theta \approx t'$;
- $\Gamma; \mathcal{R} \sqcup \mathcal{R}' \vdash_r t' : \mathrm{T}$

Proof sketch: The only rule we added to the base-level type system is rule Ver, it is therefore the only case we need to add.

Once Lemma A. is applied, the main thing left to do is to show is that $\Gamma; R \vdash \mu(s, \theta(x), k) : \mathrm{U}'$, where $\Gamma(k) = \mathrm{SSK[U']}$:

- Each condition $\theta(x) = m_i$ can be typed Bool since both term are typable by hypothesis.
- Each m_i appears in a subterm of the form $\mathsf{sign}(m_i, k)$, since k is assigned a key type, $\mathsf{sign}(m_i, k)$ can only be typed using rule Sign-Low or Sign-Msg, and in both cases, it implies m_i to have type U′.

By combining these last two points, we get that $\mu(s, \theta(x), k)$ is of the expected type, and we can conclude the proof. □

Finally, we can prove the main theorem.

Proof sketch of Theorem 3: Since the result is already known for the restricted system, we apply Lemma 17. and get an equivalent term with the same

type *in the restricted system*, and apply Theorem 2 to get the result. □

VII. Implementation & Case studies

A. Case studies

a) *Denning-Sacco signature Protocol [Bla08, DS81]:*
Continuing Example 1, we assume some agents may be corrupted and consider secrecy of the nonce N from the point of view of A and B. Let $T_0 := \mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{High}$ (resp. $T_0^d := \mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{Cst}_{\mathrm{a^d}}^{\infty} \times \mathrm{Low}$) be the type of the $\langle A, B, n\rangle$ sent to a honest agent (resp. a dishonest agent). We give type $\mathrm{SSK}\left[\mathrm{T}_0 + \mathrm{T}_0^{\mathrm{d}}\right]$ to the signature key ssk as agent A can send the first message to a honest agent as well as a dishonest agent. There are four actions to consider: two for a communication between a pair of honest agents, and two for a communication between a honest agent and a dishonest agent.

Intuitively, secrecy holds because when verifying the signature, we will gain knowledge that the message is of type $(\mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{High}) + (\mathrm{Cst}_{\mathrm{a}}^{\infty} \times \mathrm{Cst}_{\mathrm{a^d}}^{\infty} \times \mathrm{Low})$. We have two subcases:

- either we gain knowledge that the component we want to be secret has type High;
- either we learn the second component has type $\mathrm{Cst}_{\mathrm{a^d}}^{\infty}$. Therefore, when we check if this component is equal the agent name a, the test will fail and we can conclude by Eq-False-Cst.

b) *Mechanism 6 with signature (ISO-11770/part III):*
This protocol is informally described below.

$$A \to B : N_A$$
$$B \to A : \{\!|\langle B, \mathsf{sign}(A, N_A, N_B)\rangle|\!\}_{\mathrm{pkA}}$$

We consider secrecy of the pair $\langle N_A, N_B\rangle$. The situation is quite similar to the Denning-Sacco signature protocol, the key point is that the first a message is signed, and then it is encrypted with an asymmetric encryption function.

B. Implementation

This work is partially implemented in the Squirrel proof assistant, as part of the typing tactic. This tactic acts on equality in well-typed protocols; for instance, with m1 = m2 as a hypothesis, if the typing tactic successfully types m1 with type High and m2 with type Low, it derives false and immediatly concludes the goal.

Actually, the rule Ver is not implemented. Because it refers not only to the top-level subterms but to any number of subterm, implementing this rule implies changing the whole codebase, as the existing code assumes rules cannot refer to arbitrary subterms. This change seemed to masstive at the point implementation was considered in the internship.

As a simpler, but easier to implement alternative, we propose these two rules:

$$\Gamma(k) = \mathrm{SSK[T]} \quad \frac{\Gamma; R \vdash s : \mathrm{Msg} \qquad \Gamma; R \vdash t : \mathrm{High} \qquad \mathrm{T} \le \mathrm{Low}}{\Gamma; \mathrm{R} \vdash \mathsf{ver}(s, m, \mathsf{vk}(k)) : \mathrm{Cst}^0_{\mathsf{false}}} \text{ Ver-False1}$$

$$\Gamma(k) = \mathrm{SSK[T]} \quad \frac{\Gamma; R \vdash s : \mathrm{Msg} \qquad \Gamma; R \vdash t : \mathrm{Low} \qquad \mathrm{T} \le \mathrm{High}}{\Gamma; \mathrm{R} \vdash \mathsf{ver}(s, m, \mathsf{vk}(k)) : \mathrm{Cst}^0_{\mathsf{false}}} \text{ Ver-False2}$$

Fig. 3: Signature rules of the base-level system (additional, weaker but implemented)

These follow the form of all other rules i.e. each premise consist of typing a direct subterm. Because of this resemblance, the implementation was quite simple in the existing codebase. We detail the implementation:

- First, a new key type was added: SSK[·]. This change was the only one that required to modify code outside the main file of the typing tactic, since it required a new token in the lexer. When a symmetric encryption key is declared, it has to be bound to the encryption function, whereas when a signature key is declared, it has to be bound to the verification function, and not to the signature. For instance, to declare a signature key of type High, one has to write in Squirrel: name K : message, SSK[ver, High].
- Then, a function for each new rule was added. Since we chose to only implement rules of the form of the other existing rules, it did not raise any major difficulties.
- Finally, we had to chose an order in which we consider the rules i.e. which rule do we try to use on a term first. When typing a verification, one must first try to use Ver-False1 or Ver-False2 before applying rule Ver-Bool. Similarly, when typing a signature, one must first try to apply rule Sign-Low before applying Sign-Msg. In other words, we always try to type a term with the most precise type possible.

Protocol	Without dishonests		With dishonests	
	Implementation	With Ver	Implementation	With Ver
Denning-Sacco	✓	✓	✗	✓
Mechanism 6	✓	✓	✗	✓

Fig. 4: Results summary

All the case studies discussed in Section 7.1 were also tested in Squirrel, both with and without dishonest agents, and we resume our results in Fig. 4.

As expected, the rules Ver-False appear weaker than rule Ver: not all protocols we can type at hand can be typed within the Squirrel proof assistant. The problem

of these rules resides in the fact they can only learn information when the difference between the signature and the message is clear-cut: only when one is High and the other is Low (we could generalize to any pair of types that are "incompatible", namely, verifying that no terms can be simultaneously typed with both types). However, when typing a protocol with dishonest agents, the type considered is always of the form $\mathrm{T} + \mathrm{T}^{\mathrm{d}}$ with T containg a secret and T^{d} sent by the attacker, thus similar to type Low, and none of the rules Ver-False can be applied in this context.

VIII. Conclusion

We successfully added and proved new rules for signature primitives to the already existing type system for secrecy in Squirrel. These rules allowed to show weak secrecy in standardized protocols, and we corroborated at the same time the modularity of the type system, which was already displayed in the appendix on asymmetric encryption [DHL25].

Moreover, we implemented some of our rules in the proof assistant, and written files for every case studies we considered. This will enable us to test improvement of the rules easily when they are implemented in Squirrel.

A. Limitations & Future work

We here discuss limitations of the current work, potential lead to overcome them, and other future works.

a) *Full implementation:*

An obvious improvement to this work would be to implement rule Ver in the proof assistant, and check if the dishonest versions of Denning-Sacco and Mechanism 6, that are currently out of reach, can be typed within the tool.

b) *Improvement of rule Ver:*

Notice in rule Ver, in the subtree typing $C[\mathsf{true}]$, we can actually get even more information. We know the term must be of type U' because it has been signed with a key of type $\mathrm{SSK}[\mathrm{U}']$, but we also still know that it must be of type U. Therefore, ideally we could have $x : \mathrm{U} \cap \mathrm{U}'$, the difficulty resides in the fact that the intersection is not well-defined. There are some simple case where its result is obvious. For instance, $\mathrm{High} \cap \mathrm{Low}$ would be the empty type, for there is no type that is subtype of both. In this case, we could elagate the proof-tree and thus don't even require to type $C[\mathsf{true}]$.

Rules Ver-False can actually be seen as a specification of this intersection. By giving $\mathsf{ver}(s, m, \mathsf{vk}(k))$ type $\mathrm{Cst}^0_{\mathsf{true}}$ (or $\mathrm{Cst}^0_{\mathsf{false}}$), it allows to eliminate some subterms by using If-False if the verification is the condition of a $\mathsf{if} \cdot \mathsf{then} \cdot \mathsf{else}$.

c) *Towards more general expressiveness:*

As a motivating example, let us consider a simplified version of the CCIT protocol [BAN89]:

$$A \to B : \mathsf{sign}\big(B, \{\!|N|\!\}_{\mathrm{pkB}}\big)$$

Notice first that, unlike the two other protocols we've seen, we begin by encrypting a secret nonce and then we sign it before sending it. Thusfore, the type of the signature key must be SSK[Low] or equivalent, and when verifying the signature, the only knowledge we gain is that the message received is of type Low. Hence, we cannot differentiate the case where we receive a message containing a secret from the case where we just receive nonsense.

An idea to encompass this kind of behaviour is to consider *integrity*. This property would allow us to express that, when a message is signed, it could not have been modified by someone who doesn't have the signature key.

To go further in that direction, we could also consider the extension to refinement types, in order to carry predicates within the types, to be more precise about the content of a message.

For instace, currently the only types that can be given to $\mathsf{senc}(m, k, r)$ are Low and equivalent. And this type does not contain any information about the encrypted message m. All this knowledge is hidden in the type of the encryption key. We could consider the type Low{is encryption of High}.

To continue on the CCIT protocol, if these kind of refinement types were added, an analogue of rule Ver could learn that the message signed is of type $\mathrm{Cst}^0_{\mathrm{B}} \times$ Low{is encryption of High}, and this would likely allow us to prove secrecy of N.

I. Cryptographic games

$G^{\mathsf{eufcma}}_{\mathcal{A}}(1^\eta, \rho)$

$k :=_{\$} \{0,1\}^\eta$
$L_p = []$
$(m, s) \leftarrow \mathcal{A}^{\mathcal{O}_s}(1^\eta, \rho, \mathsf{vk}(k))$
if $m \notin L_p \wedge [\![\mathsf{ver}]\!]^{\mathbb{M}}(1^\eta, m, s, \mathsf{vk}(k)) = [\![\mathsf{true}]\!]^{\mathbb{M}}(1^\eta, \rho)$ **then**
  return 1
else
  return 0

$\mathcal{O}_s(m)$

return $[\![\mathsf{sign}]\!]^{\mathbb{M}}(1^\eta, m, k)$

Game 1.: Euf-Cma game for signature

References

[BC14] G. Bana and H. Comon-Lundh, "A Computationally Complete Symbolic Attacker for Equivalence Properties," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, in CCS '14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 609–620. doi: 10.1145/2660267.2660276.

[Bae+21] D. Baelde, S. Delaune, C. Jacomme, A. Koutsos, and S. Moreau, "An Interactive Prover for Protocol Verification in the Computational Model," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 537–554. doi: 10.1109/SP40001.2021.00078.

[DS81] D. E. Denning and G. M. Sacco, “Timestamps in key distribution protocols,” *Commun. ACM*, vol. 24, no. 8, pp. 533–536, Aug. 1981, doi: 10.1145/358722.358740.

[Bla08] B. Blanchet, “Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire,” 2008. [Online]. Available: https://bblanche.gitlabpages.inria.fr/publications/BlanchetHDR.html

[DHL25] S. Delaune, C. Herouard, and J. Lallemand, “Secrecy by typing in the computational model,” in *38th IEEE Computer Security Foundations Symposium (CSF 2025)*, Santa Cruz, CA, United States, Jun. 2025. [Online]. Available: https://cnrs.hal.science/hal-04666965

[BAN89] M. Burrows, M. Abadi, and R. Needham, “A logic of Authentication,” Feb. 1989.