The rules that define the operational semantics of applied pi-calculus and ProVerif are adapted from [1]. We use the constructors and destructors defined in [1] as an initial step to represent the cryptographic operations as depicted in Figure 1. We added other different constructors/destructors which are used to define our protocol. Constructors and destructors can be public or private. The public ones can be used by the adversary, which is the case when not stated otherwise. The private ones can be used only by honest participants.

Symmetric enc/dec:
Constructor: encryption of x with the shared secret key k, $senc(x, k)$
Destructor: decryption $sdec(senc(x, k), k) \rightarrow x$
Asymmetric enc/dec:
Constructor: encryption of x with the public key generation from a secret key k, $pk(k)$, $aenc(x, pk(k))$
Destructor: decryption $adec(aenc(x, pk(k)), k) \rightarrow x$
Signatures:
Constructors: signature of x with the secret key k, $sign(x, k)$
Destructors: signature verification using he public key generation from a secret key $k, pk(k), verify(sign(x, k), pk(k)) \rightarrow x$
One-way garbling function:
Constructors: garbling of x with the key $k, garble(x, k)$
Evaluation function:
Constructors: evaluation function of garbling of variables x and y with the key $k, garble(x, k)$, $evaluate(garble(x, k), garble(y, k))$
Commitment:
Constructors: committing x with the key $k, commit(x, k)$

Fig. 1. Constructors and destructors

The operational semantics used are presented in the Figure 2. A semantic configuration is a pair $\mathcal{E}$,$\mathcal{N}$,$\mathcal{L}$,$\mathcal{P}$ where the $\mathcal{E}$ is a finite set of names, $\mathcal{N}$ is sequence of messages, $\mathcal{L}$ is a set of events, and $\mathcal{P}$ is a finite multiset of closed processes. The semantics of the calculus is defined by a reduction relation $\rightarrow$ on semantic configurations, shown in Figure 2. $\mathsf{P}\{\mathcal{N}/\ x\}$ denotes that the process P is executed after replacing every occurrence of $\mathcal{N}$ with x. The process $event(M).\mathsf{P}$ executes the event $event(M)$ and then executes P. The input process $in(c, x).\mathsf{P}$ inputs a message on channel c, and executes P with x bound to the input message. The output process $out(c, N).\mathsf{P}$ outputs the message N on the channel c and then executes P.

The nil process 0 does nothing. The process $\mathsf{P}|\mathsf{Q}$ is the parallel composition of P and Q. The replication $!\mathsf{P}$ represents an unbounded number of copies of P in parallel. (**new** a)P creates a new name a and then executes P. The conditional if $M = N$ then P else Q executes P if M and N reduce to the same term at runtime; otherwise, it executes Q. Finally, let $x = M$ in P as syntactic sugar for $\mathsf{P}\{M/x\}$. As usual, we may omit an else clause when it consists of 0.

$(Nill)$ $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{0\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P}$

$(Repl)$ $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{!\mathsf{P}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}, !\mathsf{P}\}$

(Par) $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}|\mathsf{Q}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}, \mathsf{Q}\}$

(Par) $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}|\mathsf{Q}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}, \mathsf{Q}\}$

(New) $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{(\mathbf{new}\ a)\mathsf{P}\} \to \mathcal{E} \cup \{a'\}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}\{a'/a\}\}$ where $a' \notin \mathcal{E}$

(I/O) $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{out(c, M).\mathsf{Q}, in(c, x).\mathsf{P}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{Q}, \mathsf{P}\{M/x\}\}$

$(Cond1)$ $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\text{if } M = N \text{ then } \mathsf{P} \text{ else } \mathsf{Q}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}\}$ if $M = N$

$(Cond2)$ $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\text{if } M = N \text{ then } \mathsf{P} \text{ else } \mathsf{Q}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{Q}\}$ if $M \neq N$

(Let) $\mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\text{let } x = g(M_1, \ldots, M_n) \text{ in } \mathsf{P} \text{ else } \mathsf{Q}\} \to \mathcal{E}, \mathcal{N}, \mathcal{L}, \mathcal{P} \cup \{\mathsf{P}\{M'/x\}\}$
if $g(M_1, \ldots, M_n) \to M'$

Fig. 2. Operational semantics

References

1. B. Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.