
Preprint:
Ye, Kangfeng, Metere, Roberto, Woodcock, Jim et al. (2025) Formal Verification of Physical Layer Security Protocols for Next-Generation Communication Networks (extended version). [Preprint]

Formal Verification of Physical Layer Security Protocols for Next-Generation Communication Networks

Kangfeng Ye[1 [0000−0003−2460−7926]], Roberto Metere[1 [0000−0001−6992−4285]], Jim Woodcock[2,3,1 [0000−0001−7955−2702]], and Poonam Yadav[1 [0000−0003−0169−0704]]

[1] University of York, UK
{kangfeng.ye,roberto.metere,jim.woodcock,poonam.yadav}@york.ac.uk
[2] Southwest University, China
[3] Aarhus University, Denmark

Abstract. Formal verification is crucial for ensuring the robustness of security protocols against adversarial attacks. The Needham-Schroeder protocol, a foundational authentication mechanism, has been extensively studied, including its integration with Physical Layer Security (PLS) techniques such as watermarking and jamming. Recent research has used ProVerif to verify these mechanisms in terms of secrecy. However, the ProVerif-based approach limits the ability to improve understanding of security beyond verification results. To overcome these limitations, we re-model the same protocol using an Isabelle formalism that generates sound animation, enabling interactive and automated formal verification of security protocols. Our modelling and verification framework is generic and highly configurable, supporting both cryptography and PLS. For the same protocol, we have conducted a comprehensive analysis (secrecy and authenticity in four different eavesdropper locations under both passive and active attacks) using our new web interface. Our findings not only successfully reproduce and reinforce previous results on secrecy but also reveal an uncommon but expected outcome: authenticity is preserved across all examined scenarios, even in cases where secrecy is compromised. We have proposed a PLS-based Diffie-Hellman protocol that integrates watermarking and jamming, and our analysis shows that it is secure for deriving a session key with required authentication. These highlight the advantages of our novel approach, demonstrating its robustness in formally verifying security properties beyond conventional methods.
Keywords: Physical Layer Security, Formal Verification, Watermarking and Jamming, Needham-Schroeder Protocol, Diffie-Hellman Protocol, Communicating Sequential Processes, Isabelle/HOL

1 Introduction

Classic cryptographic methods [27] assume the computational hardness of inverting one-way trapdoor functions (e.g., prime factoring or discrete logarithm) and the limited computational power of adversaries. Adversaries that can use a sufficiently powerful quantum computer would employ techniques based on Shor's

algorithm [38] to break (invert) the above functions efficiently. As quantum computers are currently limited in the number of qubits required to attack deployed systems, a temporary mitigation is to use longer keys, which necessitate more complex key generation and management. This extra complexity will also need more computational power and cause delays, translating into a waste of resources and making it difficult or even impossible to be deployed in resource-constrained, power-limited, or latency-sensitive devices, typical of next-generation networks, such as 6G. Modern networks, indeed, demand enhanced security requirements [33] for highly diverse applications. An example are demands for strict quality of service (QoS), such as ultra-low latency and extreme high reliability (e.g., for extended reality and remote surgery), highly dynamic and heterogeneous wireless networks, massive number of low-cost Internet of Things (IoT) sensor devices, and evolving threats from Machine Learning (ML) and Artificial Intelligent (AI) in 6G, Physical Layer Security (PLS) [30,48] is emerging as a complementary approach to add an extra layer security. In this context, providing security guarantees at the physical layer significantly reduces the need for the above complications.

PLS aims to secure wireless communication at the physical layer (the lowest network layer) by exploiting the inherent characteristics and randomness of the wireless communication channel itself, such as fading, noise, interference, dispersion, and diversity. PLS is based on information-theoretic security or unconditional security, a paradigm of security that focuses on the fundamental limits of information leakage as defined by information theory [37]. It aims to provide security that is provably unbreakable regardless of the eavesdropper's computational resources, such as Shannon's perfect secrecy [37] , weak secrecy [42] and strong secrecy [12] (or statistical independence and asymptotically secure). PLS can resist some physical layer attacks such as spoofing, eavesdropping, and jamming (while conventional cryptography is not able to do so), can be key-less [23] (so no complex key management), and is quantum-resistant (because it is not based on computational hardness). Additionally, it has low computational complexity and latency, making it intrinsically suitable for IoT devices and heterogeneous wireless networks in 6G.

While a large amount of work in PLS focuses on secrecy, PLS techniques for achieving authentication have also received considerable attention. For example, Soderi et al.[41] proposed the watermark-based blind physical layer security (WB-PLSec) approach to achieve both secrecy and message integrity (i.e., authenticity with proper protocols). The method combines watermarking[11,24], where the sender embeds a watermark or authentication information in a content signal to create a covert channel based on a shared secret, with jamming, where the legitimate receiver selectively interferes with parts of the signal so that only it can recover the message while eavesdroppers cannot.

Notwithstanding its considerable potential, formal verification of PLS mechanisms constitutes an underdeveloped area of research. Traditional cryptographic security protocols have benefited from formal verification with useful tools such as FDR [17], Isabelle [34], Maude-NPA [15], AVISPA [2], ProVerif [8], and Tamarin-prover [3], (1) to discover attacks [26,1]; (2) to identify missing or weak assumptions [5]; (3) to propose fixes or improvements to protocols [26,5,29]; and (4) to

guarantee correctness [6,4]. These formal security analysis methods primarily focus on higher-layer cryptographic protocols. PLS protocols, however, operate in the lower-layer physical layers and use different security mechanisms from cryptography. Therefore, it is highly beneficial to formally verify PLS protocols before their vulnerabilities and missing assumptions are discovered after their deployment.

A recent work [10] has leveraged ProVerif [8], a security verification tool, to verify the secrecy of a WBPLSec-based variant (called NSWJ) of the Needham-Schroeder protocol [32]. Their verification approach, however, is carried out by formal verification experts in that specific language. *Protocol designers still cannot use the approach to inspect and verify the protocols by themselves.* In this regard, our recent work [47] provides more accessibility to formal verification through sound animation. This limits the intervention of experts and allows designers to either manually explore or automatically verify cryptographic protocols. In such a workflow, formal experts use a variant of Communicating Sequential Processes (CSP) [20,36], called the Interaction Trees [44] (ITrees) based CSP (ITrCSP) [16,46], to model security protocols in the theorem prover Isabelle/HOL [34]. Then, Isabelle can automatically generate Haskell code that, when compiled, provides a lightweight model checker or animator requiring no more expert intervention. However, this work only supports the Dolev-Yao attack model [14], the ad-hoc framework for each protocol (each protocol has a different fundamental message theory), and a terminal interface (users at least need a PC or workstation to run animators). Physical layer security is defined in terms of a different attacker model, so we extended [47] to model it.

Based on the context, this work is guided by the following research questions: **RQ1** (PLS): How to extend the ITrCSP-based framework to support PLS and its related attack model for verification? **RQ2** (Generic and generalisation) Can the ITrCSP-based framework be general to model all different attack models and security protocols in a generic message theory? **RQ3** (Case studies) What different insights can we get if we conduct a more thorough analysis of NSWJ? Can WBPLSec be used to implement the Diffie-Hellman algorithm? What benefits can PLS bring? **RQ4** (Accessibility) Can the framework be made even more accessible to a wider range of stakeholders, such as non-professional groups like students and the general public?

In this paper, we present our work that addresses these questions. Our novel contributions here are as follows: (1) a general sound animation framework to support both classic cryptography and PLS techniques, built on a generic and polymorphic message theory; (2) a more scalable implementation of intruder's message inference rules by constraining build-up messages into all the possible messages that legitimate agents can receive, instead of the naive implementation of [47] to build up all possible messages from inference rules; (3) a comprehensive analysis of NSWJ by considering not only secrecy but also authenticity in both passive and active attacks when the adversary could be within or out of the jamming ranges of each legitimate participant, to confirm the result of secrecy with [10] but contradicts the interpretation of authenticity from [10]; (4) a new WBPLSec-based DH protocol (DHWJ) to fix the man-in-the-middle attack and achieve authenticity;

(5) a new web interface to make our animation more accessible and user-friendly, in addition to the terminal interface. All definitions in this paper are mechanised and show an icon () that links to the corresponding repository artefacts[4].

The remainder of this paper is organised as follows. We review related work in Section 2, introduce sound animation and WBPLSec in Section 3, and present the modelling of security protocols in our generic framework in Section 4. And in Section 5, we describe the modelling of NSWJ and DHWJ in our framework, then present the web interface and discuss verification results in Section 6. Finally, in Section 7, we conclude and discuss future work.

2 Related work

Formal verification of security protocols. Model checking and theorem proving tools, such as Casper [25], FDR [26,17], Isabelle [34,35], Maude-NPA [15], AVISPA [2], ProVerif [8], and Tamarin-prover [3], have been commonly used to formally verify security protocols. These tools require users to have an in-depth level of knowledge in formal verification, and thus are not accessible to general protocol designers. Instead, the work presented in this paper provides a more accessible approach using sound animation, allowing protocol designers to interact, inspect, and verify protocols through interfaces, as well as to perform automated verification. Similar to our work, SPAN [9] is an animation tool and can be used by designers. But its soundness is not guaranteed.

Formal verification of PLS. Formal verification for physical-layer security protocols is a relatively new area. To the best of our knowledge, the work from Costa et al. [10] is the only one that focuses on this topic. In their work, the standard ProVerif language was extended with functions to model and analyse WBPLSec-based protocols symbolically. Furthermore, an attack model for WBPLSec was implemented, and their work demonstrated that NSWJ with WBPLSec is secure (in terms of secrecy) against active attackers within the jamming range and insecure otherwise. Their work formally demonstrates the sufficiency of watermarking and jamming to ensure secrecy and authenticity. Inspired by [10], our work verifies WBPLSec and its based NSWJ protocol, in the same scenarios and under the same common principles, such as its attack model and assumptions. Conversely, our work goes beyond to comprehensively analyse all *four scenarios*, where the eavesdropper is located at different positions in terms of Alice's and Bob's jamming ranges, under both *passive and active attacks*. The analysis provides previously undiscovered results: *authenticity is preserved even when the eavesdropper is outside the jamming range (and thus, the loss of secrecy)*. We also propose a new variant (DHWJ) of DH, based on WBPLSec, and conduct a comprehensive analysis of it. The other benefit of our work, if compared to theirs, is the *accessibility* of verification to protocol designers. It is worth noting that *the capability to understand and debug protocols* is crucial for protocol designers too,

[4] We assume basic knowledge of Isabelle/HOL from interested readers to understand these definitions and theorems.

Table 1. Comparison of related work where ○ for no, ● for yes, and * for comments.

Related work	Primitives		Attack		Sound	Verification				UI	
	Crypt	PLS	DY	PLS		DBG	UGV	Reach	Feas	Term	Web
Tamarin	●	○	●	○	●*	○	○	●	○	●	●*
ProVerif	●	○	●	○	●*	○	○	●	○	●	○
Costa et al. [10]	●	●	●	●	●*	○	○	●	○	●	○
SPAN [9]	●	○	●	○	○	●	○	●	○	●	●
Ye et al. [47]	●	○	●	○	●	●	●	●	●	●	○
This work	●	●	●	●	●	●	●	●	●	●	●

Acronym Crypt: cryptography; Attack: attack models; DY: Dolev-Yao; DBG: debugging; Reach: reachability check; Feas: feasibility check; Term: terminal interface; UGV: user-guided verification.

especially in new areas such as PLS. As a result, we demonstrate how our analysis facilitates a deeper understanding of the protocol, as it contradicts one of the conclusions in [10], as mentioned previously.

Sound animation. For another comparison, ITrCSP has been applied to animate robotic control software [46] modelled in RoboChart [31,45] and security protocols [47], and demonstrate functional correctness through manual interaction and automatic checking. Our work is also based on ITrCSP and extends [47] in three aspects: a general modelling framework, the WBPLSec security protocol, and a web user interface. The sound animation approach presented in [47] primarily focuses on two case studies. Its modelling framework in ITrCSP and Isabelle is not generic. Thus, every protocol has an ad-hoc message theory in Isabelle. Our work presents a generic and polymorphic message theory, enabling all protocols to share a common message framework. Additionally, each protocol can be instantiated for specific configurations. This will be discussed in Section 4. We also extend the modelling framework to support a new attack model for WBPLSec in addition to the Dolev-Yao model. WBPLSec-based protocols, such as NSWJ and DHWJ, can be modelled and verified in the same framework. From an interface aspect, we develop a new web interface to allow users to easily configure properties, verify, and view their interactions with models and counterexamples.

Summary. Table 1 summarises the comparison of this work with others in modelling primitives and attack models, soundness, verification, and interfaces. It is worth noting that (1) the claimed soundness in Taramin and ProVerif is not formally verified, unlike our work where the soundness is guaranteed by Isabelle/HOL; and (2) Taramin has a web interface for its interactive proof mode, but not intended for designers.

3 Background

ITrCSP and Sound animation. Animation models interactions of reactive systems with their environment through events. We utilise Interaction Trees [44] (ITrees), which allow for a formal specification to be associated with both abstract and executable denotational semantics [43], as coinductive tree structures. This en-

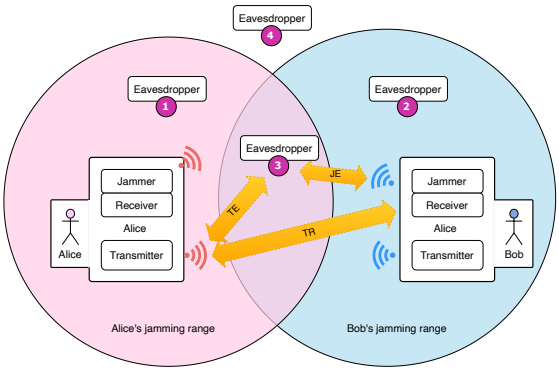


Fig. 1. An illustration of WBPLSec with two legitimate agents (Alice and Bob) and an eavesdropper in different spatial locations in terms of the agents' jamming ranges. There are four combinations of locations, denoted as Eve1, Eve2, Eve3, and Eve4.

ables the construction of denotational and operational semantics, facilitating both reasoning and implementation. Using ITrees, Foster et al. [16,46] gave semantics to a deterministic version of CSP process algebra [20,36], called ITree-based CSP or ITrCSP, and mechanised it in Isabelle/HOL [34] to allow automatic generation of Haskell code from a CSP model for animation. The basic processes and operators of ITrCSP are summarised in [47, Table 1]. Further details about their definitions, semantics, and implementation in Isabelle/HOL can be found in [16,46].

The animation is sound thanks to ITree's executable semantics and Isabelle's code generator [19], which translates executable ITree definitions in HOL logic to target functional languages (such as Haskell). Soundness is not a formally proved property here, but the simple and unbroken link from operational semantics of CSP, to (corecursive) definitions of CSP operators in ITrees, and the final Haskell code through an equational logic-based code generator (all in Isabelle/HOL) ensures its soundness. We illustrate this link in Figure 4 of Appendix A.

PLS: Watermarking and Jamming. Soderi et al. [41] proposed WBPLSec, which combines watermarking [11,24] with jamming (based on iJAM [18] but improved the data rate from half to full) for secure communication. During the transmission, the source clear message and the watermarked signal (from a part of the source message) are transmitted in two independent paths, either embedded both signals in a narrow-band channel [41] or two different visible lights (the blue light and the red light) in visible light communication (VLC) [40]. Figure 1 illustrates the concept of WBPLSec schematically. We consider two legitimate agents, Alice and Bob, and one eavesdropper (Eve) in various locations (Eve 1, 2, 3, and 4) in terms of the jamming ranges (the two large circles in the diagram) of Alice and Bob. Each of the legitimate agents is equipped with a transmitter and a receiver which contains a jammer. There are three kinds of links depicted in the diagram: TR for the transmitter-receiver link, TE for the transmitter-eavesdropper link, and JE for the jammer-eavesdropper link. Considering a scenario where Alice wants to send a confidential message m to Bob, the WBPLSec approach works as follows:

S1. the transmitter of Alice modulates m to get the host regulated signal x_S of length N, of which the first N_w samples x_w are selected to use direct sequence spread spectrum (DSSS) for watermarking (by using the pseudo-noise PN code c_w to spread the x_w to decorrelate the host signal with the watermark later) to get the spread-spectrum (SS) watermark w;
S2. the transmitter embeds w in x_S to get x'_S and sends it;
S3. then the receiver Bob uses his jammer to jam M (where $M \leq N_w$) samples over x'_S to get an intentionally interfered and corrupt $\hat{x}_S$;
S4. Bob can extract the watermark $\hat{x}_w$ from the received signal (via the TR link) by using an additional DSSS demodulator based on the same PN code c_w;
S5. Bob's receiver then removes corrupted jammed samples from $\hat{x}_S$ and replaces them with unjammed samples from $\hat{x}_w$;
S6. if Eve, such as Eve 2, is within the jamming range of Bob, Eve is not able to construct clean messages from $\hat{x}_S$ (received through the JE link) because she does not know c_w;
S7. otherwise, if Eve is out of the jamming range of Bob, she can receive the watermarked signal x'_S through the TE link and understand the clean message x_S.

To ensure secrecy and integrity, Alice and Bob share a common secret, N and c_w, beforehand and never send the secret over the network. Bob will use another PN code for his transmitter, which Alice is also aware of in advance.

The security of WBPLSec-based protocols is highly dependent on the location of the eavesdropper. To minimise the information leakage, the location of legitimate agents, their transmission and jamming powers need to be considered. One possible solution is to ensure that jamming fully covers the transmission area. This is feasible if the receiver (such as an edge node) is more powerful or visible light in VLC is restricted in an area (for example, by walls). Another solution is to use reconfigurable intelligent surfaces (RIS) to dynamically direct transmission signals towards legitimate receivers and then direct jamming signals to cover that transmission area, thereby forming a secure region [39].

4 Modelling of Security Protocols

Our framework supports the symbolic modelling of both classic cryptographic and PLS protocols. The formalization of cryptography and the Dolev-Yao attack model [14] have already been described in [47]. For this reason, this section presents our formalization of PLS protocols in excruciating detail, instead of cryptography.

Assumptions. For WBPLSec, we assume

AS1. As a sender, each legitimate agent has a unique secret code used for watermarking.
AS2. As a receiver, each legitimate agent knows the unique secret codes which the corresponding senders use for watermarking, and the agent then uses the code to jam the communication from these senders.

AS3. The intruder (or Eve)[5] does not know or guess the secret codes that legitimate agents use for watermarking and jamming.
AS4. The intruder may have her code for watermarking, but her code is not known to any legitimate agents.
AS5. These secret codes are never transmitted over the network.

Network model. A network model describes how different agents and the intruder communicate through proper (secure or public) channels. For WBPLSec, we employ a different network model than that used for cryptography [47] because watermarking and jamming differ from conventional cryptography: they establish a secure region (based on locations), whereas cryptography enforces security through encryption, hashing, and other primitives. In the new network model, each legitimate agent sends a watermarked message $wm = \mathsf{wat}(m, bA_w)$ using its watermarking bitmask bA_w (corresponding to the secret code, as with [10]). The intruder has no jammer because she always tends to hear more information instead of interfering, and will receive wm. Its receiver relays wm to Bob, and at the same time, it sends a jammed message $jm = \mathsf{jam}(wm, bB_j)$ (using Bob's jamming bitmask bB_j) to the main body of the intruder if the intruder is within the jamming range of Bob. Otherwise, the receiver sends wm to the intruder. This way, the intruder will hear the jammed and watermarked message or just the watermarked message, depending on her location. More details will be given later when we discuss the modelling of NSWJ using Figure 3.

Attack models and inference rules. In WBPLSec, Eve is usually passive. She captures transmitted signals and tries to learn more knowledge from them. In this paper, we also consider an active Eve who can send fake messages to legitimate agents. The information she can derive and the messages she can convey are controlled by inference rules based on her current knowledge. Such rules are defined in Table 2 where K, a set of messages, denotes Eve's current knowledge. Rules are divided into two groups: *breakdown* rules, where $(K \vdash_{\Downarrow} m)$ denotes a message m can be derived from K using one of member, unpairing, decryption, digital verify, watermarking, and jamming rules; and *build-up* rules, where $(K \vdash_{\Uparrow} m)$ denotes a message m can be derived from K using one of member, pairing, encryption, digital sign, and watermarking rules. We note that the Dolev-Yao cryptographic inference rules are also supported and included in the table to make our inference more general.

As with [10], $\mathsf{wat}(m,n) = \mathsf{wat}(m',n')$ only if $m = m'$ and $n = n'$. That means the same watermarked messages can be generated only if watermarking applies to the same message using the same bitmask. Also, $\mathsf{jam}(m,0) = m$ denotes that jamming using an empty bitmask is like no jamming. Particularly, the jamming rule defines that if jamming a watermarked message using the prefixing bitmask b' of the watermarking bitmask b, the watermarked message can be recovered when b' is known. Here, we treat a bitmask as a sequence of bits and use the sequence concatenation $\frown$ to establish that b' is the prefix of b. This corresponds to M and N_w in S3.

Informally, if the intruder sits within the jamming range of a receiver A, she can only hear jammed watermarked messages. According to AS3 and the inference

[5] We use *Intruder* in the modelling of security protocols and only use *Eve* when discussing WBPLSec in general.

Table 2. Intruder message inference rules: breakdown and build-up rules.

Name	Id	Premises	Break down	Build up
Member	Mb	$m \in K$	$K \vdash_{\Downarrow} m$	$K \vdash_{\Uparrow} m$
Pairing	Pa	$m \in K \wedge m' \in K$		$K \vdash_{\Uparrow} (m,m')$
Unpairing	Up	$(m,m') \in K$	$K \vdash_{\Downarrow} m \wedge K \vdash_{\Downarrow} m'$	
Enc/Sign	Enc	$m \in K \wedge k \in K$		$K \vdash_{\Uparrow} \{m\}_k$
Dec/Verify	Dec	$\{m\}_k \in K \wedge k^{-1} \in K$	$K \vdash_{\Downarrow} m$	
Water-	Wat1	$\mathsf{wat}(m,b) \in K$	$K \vdash_{\Downarrow} m$	
marking	Wat2	$m \in K \wedge b \in K$		$K \vdash_{\Uparrow} \mathsf{wat}(m,b)$
Jamming	Jam	$\mathsf{jam}(\mathsf{wat}(m,b),b') \in K \wedge$ $b' \in K \wedge \exists x \bullet b' \frown x = b$	$K \vdash_{\Downarrow} \mathsf{wat}(m,b)$	

rule Jam in Table 2, she cannot derive the watermarked messages from the learned messages. So, secrecy is preserved. However, if she sits outside the jamming range but still within the transmission range of the sender, she may still be able to hear watermarked messages. According to the Wat1 inference rule, she can derive clear messages from the watermarked messages. So, secrecy is violated. According to AS3, in both cases, message integrity is maintained because the intruder cannot forge a watermarked message without knowledge of its secret code. An active intruder can build up all messages from her knowledge using the build-up rules. And then she can fork or send these (possibly watermarked using her secret code) messages to the network. According to AS4, the messages are neither expected nor accepted by legitimate agents because the secret code is unknown to other agents. Indeed, there is little difference between passive and active attack models.

Message data types. We define a generic message type which is parametric in the sizes of various entities in messages, such as agents, nonces, keys, bitmasks, and modulo exponent bases. We require these entities to be finite for the sake of the executable nature of animation. This generic type is achieved through a data type `fsnat`, defined below, for a finite set of natural numbers. Consequently, each entity (from a finite set of n entities) can be numbered from 0 to $n-1$.

```
typedef (overloaded) ('n::len) fsnat = "0..<LENGTH('n)::nat set"
```

The `fsnat` is a polymorphic type with one type variable `'n` of class `len`. The `len` class is equipped with a `LENGTH` function to get its size. So the new type `'n fsnat` is isomorphic to a finite set of natural numbers from 0 to the size of `'n` minus 1. This type comes with a pair of functions `nat_of_fsnat` and `fsnat_of_nat` to convert from `fsnat` to a natural number or from a natural number to a `fsnat`. To facilitate the construction of such a `'n fsnat` value, we define `nmk` ( ) function to construct a `'n fsnat` value from its input natural number `x`. Then a variety of data types can be defined using `fsnat`.

```
datatype ('n::len) dagent = Agent (ag:"'n fsnat") |Intruder |Server
datatype ('k::len,'s::len) dkey = Kp "fsnat['k]" | Ks "fsnat['s]"
datatype ('bm, 'bl) dbitmask = Null | Bm  "'bm fsnat" "'bl fsnat"
```

Participants or agents are modelled using `'n dagent` where `'n` is the number of legitimate `Agent`s except an `Intruder` and a `Server`. In the definition, `ag` is a function to extract the agent number from an `Agent`. If such a function is applied to `Intruder` or `Server`, an exception will be raised in the generated code. Nonces (of type `dnonce`) and modulo exponent bases (`dexprg`) are just `fsnat` and their definitions are omitted here. Keys (of `dkey`) are either public by the constructor `Kp` or private by the constructor `Ks`.

Bitmasks (of type `dbitmask`) contain an additional `Null` for an empty bitmask. The type has two type variables `'bm` and `'bl` to represent the number of bitstrings and the maximum length of a bitstring used for watermarking or jamming. Additionally, this data type is instantiated to partial `order`, so any two bitmasks can be compared using $\leq$. And two bitmasks `bm1` $\leq$ `bm2` if either (1) `bm1` is empty, or (2) both `bm1` and `bm2` are not empty (such as `Bm b1 l1` and `Bm b2 l2`), their bitstrings are equal (`b1=b2`), and their lengths are less than or equal (`l1`$\leq$`l2`). This is to implement the abstract prefix relation in Table 2 to derive the watermarked message from its jammed counterpart.

Based on these types, a message type is defined below where only a few constructors are shown and the complete definition is given in Appendix B.

```
datatype ('a, 'n, 'k, 's, 'g, 'bm, 'bl::len) dmsg = ...
 | MBitm  (mbm:"('bm, 'bl) dbitmask")                    b_{m}_ _
 | MWat   (mwm:"T dmsg") (mwb:"T dmsg")                  {|_|}^{w}_ _
 | MJam   (mjm:"T dmsg") (mjb:"T dmsg")                  {|_|}^{j}_ _
```

All messages for communication on channels are of the polymorphic type `dmsg` with 6 type variables to represent the numbers of agents (`'a`), nonces (`'n`), public keys (`'k`), private keys (`'s`), modulo exponent bases (`'g`), and bitmasks (`'b`). `T` in the definition above is a shorthand for `('a, 'n, 'k, 's, 'g, 'bm, 'bl::len)`. Messages can be agent's identities of type `dagent` (with the constructor `MAg`), nonces (with `MNon`), public and private keys (with `MK`), pairing of two messages (with `MPair`), modulo exponent bases (with `MExpg`), modulo exponentiation (with `MModExp`), bitmasks (with `MBitm`), watermarking (with `MWat`), and jamming (with `MJam`). To simplify and shortening formulas, we introduce the notations $\langle m_1, m_2 \rangle$, $g_m n$, m^e, $b_m nl$, $\{\!|m|\!\}^w_b$, and $\{\!|m|\!\}^j_b$ to denote them respectively.

Message inferences. In Isabelle, we define a function `breakm` (🌼) for the breakdown rules in Table 2 to derive a list of messages from a given list for the sake of implementation in Haskell. The `breakm` is defined on another function `break_lst` (🌼). We omit the details of these functions here for simplicity.

For the build-up rules, [47] uses `buildm` to construct all possible messages by these rules. The number of these messages is usually huge. It is time-consuming and space-consuming, and has a significant impact on performance. To address this issue, we propose a solution that considers only the possible acceptable messages from all legitimate agents and disregards other unacceptable messages. This is a reasonable assumption because the transmission of these unacceptable messages will not change the behaviour of all legitimate agents.

For the approach, we first define a function `buildable m ms` (🌼) to return whether a message `m` can be constructed from a set `ms` of messages using the

build-up rules. Then `filter_buildable xs ms` () returns a list of messages, of which each message is from `xs` and buildable from `ms`. This improves the inference performance dramatically in animation.

Generally, when the intruder hears a new message, she attempts to break it down using her existing knowledge and updates it accordingly. Then, if modelling an active attack, she will construct all buildable messages from her up-to-date knowledge using the `filter_buildable` function and send them to the network.

Signals. We declare signals to specify properties. The definition `dsig` () is the same as [47] and omitted here.

Channels. Channels for communication are grouped in a definition `chan` below, where seven channels (with their corresponding message types) are declared.

```
datatype T chan =
 env  :: "'a dagent×'a dagent"
 send :: "'a dagent×'a dagent×'a dagent×T dmsg"
 recv :: "'a dagent×'a dagent×'a dagent×T dmsg"
 cjam :: "T dmsg"   sig  :: "('a, 'n) dsig"
 leak :: "T dmsg"   terminate :: "unit"
```

We extend the traditional channels `send` and `recv` in [47] with an additional component such as `M` in `send!(A, M, B, m)` to denote the medium through which the message is sent or received. If `M` is the `Intruder`, then it denotes a message through a public and insecure channel. Otherwise, it implies the use of a private channel (so the intruder cannot hear such a message). With this additional component, we can model both public and private channels. And `hear` and `fake` (or `relay`) are just synonyms to `send` and `recv`. The channel `cjam` relates to jamming communications. The channels `sig`, `leak`, and `terminate` are used to signal the stages which the protocol is on. Eventually, they are used to specifying security protocols. For example, secrecy is specified using `leak` events and authenticity is specified through signals `StartProt` and `EndProt` on channel `sig`. The `terminate` event indicates a successful completion of a protocol run.

In brief, our formalization establishes structured inference rules to determine what an intruder can know or do, while models, watermarking, and jamming restrict their capabilities through physical-layer security techniques.

5 Needham-Schroeder and Diffie-Hellman Protocols

The Needham-Schroeder protocol [32] is used to establish mutual authentication between two parties (such as Alice and Bob) over an insecure or public network, utilizing either symmetric or asymmetric encryption. In this paper, we consider a variant of NSPK, referred to as NSWJ here, which utilizes watermarking and jamming (instead of asymmetric encryption) to achieve secrecy and authenticity, similar to [10]. We illustrate the message exchange between Alice and Bob for both NSPK and NSWJ below, where the asymmetric encryption in NSPK is replaced

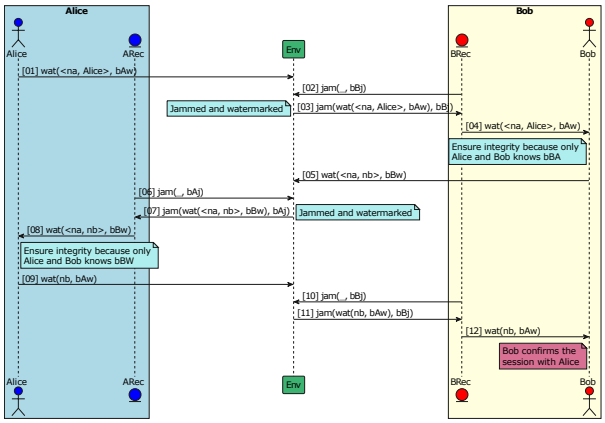


Fig. 2. The messages between Alice and Bob for NSWJ.

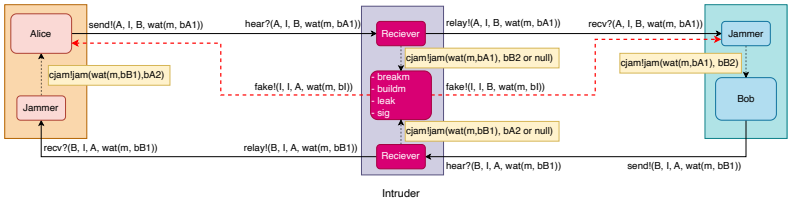


Fig. 3. The modelling of NSWJ in ITree-based CSP.

by watermarking in NSWJ.

$$\begin{array}{cc} NSPK & NSWJ \\ A \to B : \{\!|(na, Alice)|\!\}^{a}_{pk_B} & A \to B : \{\!|(na, Alice)|\!\}^{w}_{bA_w} \\ B \to A : \{\!|(na, nb)|\!\}^{a}_{pk_A} & B \to A : \{\!|(na, nb)|\!\}^{w}_{bB_w} \\ A \to B : \{\!|nb|\!\}^{a}_{pk_B} & A \to B : \{\!|nb|\!\}^{w}_{bA_w} \end{array}$$

We further illustrate the detailed message exchange in NSWJ in Figure 2, where ARec and BRec represent the receivers of Alice and Bob, respectively. Alice starts the session by sending a watermarked message (01) to the environment Env (i.e., the secure region) containing a nonce `na` and her identity, using her watermarking bitmask. BRec starts to jam (02) Alice's watermarked message using Bob's jamming bitmask and receives the (03) jammed watermarked message from the environment. BRec can recover the watermarked message because BRec knows the jamming code and sends it (04) to Bob. Bob can confirm this message is from Alice because it is watermarked using Alice's bitmask. Afterwards, Bob sends (05) back Alice's nonce `na` and his nonce `nb` to Env. Similarly, ARec will jam (06) the message, receive (07) the jammed and watermarked message, recover the watermarked message from Bob, and send (08) it to Alice. Alice confirms the integrity of the message and sends (09) `nb` back to Bob. BRec will jam (10), receive (11), recover, and send (12) it to Bob. Finally, Bob can confirm he has established a secure session with Alice if Eve is only within the jamming range of Alice and Bob.

Figure 3 schematically depicts the modelling of NSWJ in ITrCSP. We model Alice, Bob, and the intruder as three processes. Both Alice and Bob contain a jamming receiver (or jammer), and the intruder includes a receiver (Figure 3 splits the receiver into two elements for better visualisation of connections). Alice and Bob `send` messages to the intruder and receive messages on the `recv` channel from the intruder. The intruder `hear`s messages that are sent by Alice or Bob and can `fake` messages (if he is an active attacker) that Alice and Bob will receive. The receiver of the intruder will relay the received watermarked messages to intended agents (such as Bob), and jam the messages using the bitmasks (such as `bB2`) of their intended agents (or a `Null` bitmask) if he is in the jamming range (or out of the range) and send the jammed messages to the intruder through the `cjam` channel. While the intended agents receive the relayed watermarked messages, the jammer jams the messages using his jamming bitmask (such as `bB2`) and sends the messages to the agents (such as Bob). This way, we can flexibly model the location of the intruder, and his location will not affect the message Bob receives, because the intruder's receiver will jam the watermarked messages accordingly based on his location. We note that the `cjam` channel is represented by a dashed line, indicating that jamming is an internal behaviour and will be hidden from the modelled protocol in CSP.

The main body of the intruder, represented as a box in the middle, describes its main behaviour: breaking down heard messages, building up new fake messages, and sending them to the network (if it is active), leaking secrets if the secrets are within its knowledge, or dealing with signals.

To model the protocol in Isabelle using ITrCSP, we first define a *configuration* of this protocol, which includes the instantiation of message types as follows.

```
type_synonym max_agents = 2; max_nonces = 4; max_bm = 3;
  max_bm_len = 2; max_pks = 1; max_sks = 1; max_expg = 1
type_synonym dagent = "max_agents dagent"
```

First, we define type synonyms, such as `max_agents`, as a numeral type 2 and use these synonyms to instantiate polymorphic data types, `dagent`, etc. We give the same names to type synonyms of these concrete types. When we refer to these types in the rest of the paper, we mean the specific types mentioned in this section. In this protocol, we define three bitmasks, each with a length of up to 2 (`max_bm_len`). The protocol is configured below.

```
abbreviation "Alice ≡ Agent (nmk 0) :: dagent"
abbreviation "Bob ≡ Agent (nmk 1) :: dagent"
abbreviation "nonmap ≡ {Alice↦ nmk 0, Bob↦nmk 1, Intruder↦nmk 2}"
abbreviation "bmmap ≡ {Alice↦Bm (nmk 0) (nmk 1),
 Bob↦Bm (nmk 1) (nmk 1), Intruder↦Bm (nmk 2) (nmk 1)}"
```

Such a configuration maps the legitimate agents Alice and Bob to the agent, nonce, and bitmask indices, where the constructors like `nmk` are used to make an instance of `fsnat`. The location of an intruder is defined in terms of jamming ranges and configured using the map below.

```
datatype deve = Eve1 | Eve2 | Eve3 | Eve4
abbreviation "evemap eve ≡ case eve of
  Eve1 ⇒ {Alice↦T, Bob↦F} | Eve2 ⇒ {Alice↦F, Bob↦T} |
```

$$\texttt{Eve3} \Rightarrow \{\texttt{Alice} \mapsto \texttt{T}, \texttt{Bob} \mapsto \texttt{T}\} \mid \texttt{Eve4} \Rightarrow \{\texttt{Alice} \mapsto \texttt{F}, \texttt{Bob} \mapsto \texttt{F}\}$$

The `evemap` configures the location (among four, see Figure 1) of the intruder (`eve`) based on where the intruder is within the jamming range of Alice and Bob (where `T` for True and `F` for False). Then we use these configurations to define the jamming process (`jamming`) for Alice and Bob, the jamming process (`jammingI`) for the intruder, the process (`PAlice`) for Alice, the process (`PBob`) for Bob, and the process (`PIntruder`) for Intruder where her initial knowledge is taken into account. We discuss the definition of `PAlice` more below and omit the details of other processes here for simplicity.

$$PAlice \mathrel{\widehat{=}} \big(\big(\, Initiator(\_) \,\|_{jevents}\, jamming(\_)\,\big) \setminus jevents\big) [\![\{ Terminate \} \triangleright skip$$

`PAlice` is a parallel composition of an `Initiator` (see [47] for details) for the behaviour of Alice, and `jamming` over the jamming events (`jevents`) with these events hidden. This process can be terminated using the CSP exception operator over the `Terminate` channel. Finally, the protocol NSWJ is composed of the processes for Alice, Bob, and the intruder in response to appropriate events.

$$\texttt{NSWJ} \mathrel{\widehat{=}} (PAlice \,\|_{TermEvent}\, PBob) \,\|_{ABIEvents}\, PIntruder$$

This is the process used to generate an animator for the protocol.

The Diffie-Hellman (DH) [13] protocol aims to establish a shared secret between two agents using agreed-upon information over an insecure network. It is commonly used to derive new session keys. DH is based on modular exponentiation $g^a \bmod p$ where g is the base, a is the power or exponent, and p is the modulus. In [47], we model the DH with three messages: (1) $A \rightarrow B : g^{na}$ and B computes $k_B = g^{na^{nb}}$, (2) $B \rightarrow A : g^{nb}$ and A computes $k_A = g^{nb^{na}}$, (3) $A \rightarrow B : \{\!|t|\!\}^s_{k_A}$ where t is a secret and the superscript s denotes a symmetric encryption, and B decrypts it using k_B to get t. Using sound animation, it is easy to demonstrate a man-in-the-middle attack where secrecy is not preserved.

Next, we consider a variant of DH, called DHWJ, which uses watermarking and jamming. It is composed of four messages: (1) $A \rightarrow B : \{\!|g^{na}|\!\}^w_{bA_w}$ and B computes $k_B = g^{na^{nb}}$, (2) $B \rightarrow A : \{\!|g^{nb}|\!\}^w_{bB_w}$ and A computes $k_A = g^{nb^{na}}$, (3) $A \rightarrow B : \{\!|\{\!|t|\!\}^s_{k_A}|\!\}^w_{bA_w}$ and B decrypts it using k_B to get t, (4) $B \rightarrow A : \{\!|\{\!|t|\!\}^s_{k_B}|\!\}^w_{bB_w}$ and A decrypts it to confirm B knows t. The message exchange and protocol configuration () are similar to those of NSWJ and are omitted here for simplicity.

6 Verification and Evaluation

In Isabelle, we can automatically generate the Haskell code for the NSWJ and DHWJ protocols. To facilitate the automatic or manual exploration of the design to check secrecy and authenticity, we developed a new web interface[6] in addition to the terminal interface (as presented in [47]) to directly visualise sequence diagrams of animation and counterexamples (attacks), as illustrated in Appendix C.

[6] The source code is available at https://github.com/RandallYe/Animation_of_Security_Protocols/tree/master/animation-web-ui

Table 3. Comparison of verification results: NSPK vs. NSWJ, and DHWJ vs. DH in the presence of an active intruder, except the specific NSWJ† for a passive intruder.

Properties	NSPK	NSWJ				NSWJ Passive†				DH	DHWJ			
		E1	E2	E3	E4	E1	E2	E3	E4		E1	E2	E3	E4
Secrecy	○	○	○	●	○	○	○	●	○	○	○	○	●	○
Authenticity for Alice	●	●	●	●	●	●	●	●	●	○	●	●	●	●
Authenticity for Bob	○	●	●	●	●	●	●	●	●	○	●	●	●	●

The verification algorithm underneath interfaces is the depth-first exploration of the event space up to specified steps for a protocol design. The terminal interface provides users with manual exploration, automatically exhaustive or random reachability check in terms of trace properties, or feasibility check of a specified sequence of events or trace. Random exploration and feasibility check are not yet implemented in the web interface. It, however, supports sessions and enables designers to (simultaneously) manually explore or exhaustively check reachability and view counterexamples visually in sequence diagrams, as shown in Figure 2. Graphical visualisation of protocols as sequence diagrams has been demonstrated in [28]; however, their approach is design-oriented rather than analysis-oriented, and their diagrams are manually created, whereas our diagrams are dynamically updated. We could not extend it, as manually created diagrams lack the precise execution semantics needed for automated visualisation and attack simulation.

While the web interface provides no more verification features than the terminal interface functionally, the diagrammatic view of counterexamples helps designers easily follow their interactions with animators and identify problems. Additionally, verification is more easily configured through the web interface, allowing for the selection of intruder location and properties to check. For both protocols, the web interface will enable users to select which location of Eve to animate and verify, manually animate the protocol, automatically verify secrecy and correspondence properties such as authentication, or combine manual exploration with an automatic check for user-guided verification. More details about the web interface can be found in Appendix C.

Using the interfaces, we analysed NSWJ and DHWJ, along with their cryptographic counterparts, NSPK3 and DH, in terms of secrecy and authenticity. Table 3 shows verification results in the presence of an active intruder and NSWJ in the presence of a passive intruder. The results of NSPK and DH are already discussed in [47]. It is not surprising that secrecy is only satisfied in the scenario of Eve3 (E3 in the table) because Eve can only hear jammed messages from both Alice and Bob (so she cannot derive clear messages). Authenticity for both Alice and Bob is preserved when using WBPLSec due to the watermarking feature. We note that DH is not an authentication protocol and usually other mechanisms such as digital signature, hash, trusted certificate authority (CA), and message authentication code (MAC) are required to achieve both secrecy and authenticity such as in the Unified model and MQV [7], SIGMA [21] and HMQV [22]. With PLS, we demonstrate that both properties can be achieved without such a mechanism, regardless of Eve's location.

While the ProVerif analysis in [10] uses an active intruder, we consider both active and passive intruders. Our animation shows that whether the intruder is active or passive makes no difference in terms of secrecy and authenticity. If a property is held in the passive model, it is also held in the active model and vice versa. This is related to AS4 where neither Alice nor Bob knows the Intruder's watermarking bitmask, and therefore neither Alice nor Bob will accept the watermarked (inferred) message from the Intruder. This should not be surprising because watermarking can ensure message integrity.

Although the result of NSWJ in [10] is correct and consistent with ours, we argue that their speculation "WBPLSec, in which the receiver, besides the sender, also plays an active role in the communication, so keeping authenticity between the enrolled parties" is not accurate. It is recognized that the real problem with NSPK is its second message $\langle na,nb\rangle$, sent by Bob to Alice, which contains no information about Bob's identity (authenticity is not guaranteed). This permits a man-in-the-middle to forward such a message to Alice, making her believe it originated from the intruder (while Bob generated it). To support that, the fix [26] appends Bob's identity to the second message. We conclude that NSWJ ensures authenticity not due to jamming, but instead to watermarking, thanks to its implicit integrity guarantee. This evidence suggests that our formalism can enhance the understanding of protocols and their security mechanisms for designers.

To evaluate performance, we ran all the verification on the same MacBook Pro laptop. For both the original NSPK and its corrected version [26], it took about 1 second (s) in Tamarin and about 200 milliseconds (ms) in ProVerif to verify all secrecy and authenticity properties. It took about 4 seconds in our tool for each property. For the NSWJ version in [10], ProVerif uses 186 ms to prove four secrecy properties when considering Eve3. For the same protocol, our tool uses less than 1 second to verify each secrecy or authenticity property when considering the same eavesdropper location Eve3. However, it will take about 7 seconds to verify each secrecy property when considering other eavesdropper locations and about 9 (or 33) seconds to verify an authenticity property when considering Eve1 or Eve2 (or Eve4). So, it takes longer for our tool to verify properties, which is due to the executable nature of our approach to achieve accessibility for designers. The verification of the Diffie-Hellman protocols, including DH and DHWJ is similar.

In a nutshell, we successfully address all research questions. We extended our framework to support both the Dolev-Yao and PLS attack models, and verified two PLS-based security protocols (**RQ1**). We generalised a message theory to be generic. So security protocols using different primitives and attack models can be modelled and verified in the same framework (**RQ2**). We proposed DHWJ. With a thorough analysis of NSWJ and DHWJ using our approach, we found that authenticity is due to the watermarking (as we expect) and is preserved in all scenarios, even when secrecy is compromised. That is one benefit that PLS can bring (**RQ3**). We developed a web interface and used it to verify both the cryptographic-based and PLS-based WJ and DH protocols. Indeed, this can be carried out by a wide range of stakeholders (**RQ4**).

7 Conclusion and Future Work

This paper presents a formal verification framework for Physical Layer Security protocols based on sound animation with graphical interface to improve accessibility. Our work builds upon the formalism by [47], based on ITrCSP and implemented in Isabelle/HOL. We further extend its generality by supporting both cryptographic and PLS protocols, by improving its intruder's message inference rules, and by introducing a new web interface. We apply our framework to model and verify the WBPLSec-based Needham-Schroeder protocol proposed in [10]. We further integrate watermarking and jamming techniques into the Diffie-Hellman protocol, and our analysis shows that the proposed protocol preserves both secrecy and authenticity when a new session key is derived.

Our work has limitations. We verify protocols through the exploration of event trees using animators. The event trees could be too big to be exhaustively explored for complex security protocols. For the same reason, our tool needs a longer time to verify these protocols. So, scalability is still a challenge. Our current approach reduces the intervention from formal experts, which is still needed to model security protocols in Isabelle initially. They need to write CSP processes for each legitimate agent and a process for the intruder, consider what messages these processes can send and receive, and then compose the processes together through corresponding events. With significant additional effort, improvements would require to (1) design or utilise a domain-specific language (DSL) by designers to capture their protocols, properties, and attack models; (2) automatically transform the DSL models to ITrCSP in Isabelle; and (3) automatically generate animators.

Following [10], we consider a simplified scenario where the jamming signal remains consistent within a given distance, and no signal is received beyond this distance. In reality, the jamming signal and transmission signal are degraded as the distance increases. We use this assumption to simplify scenarios in terms of the eavesdropper's location: either within the jamming range or outside the range. One of our future projects is to consider more complex scenarios.

Future work will also focus on verifying more cryptographic protocols, such as the Mesh Commissioning Protocol (MeshCoP) for the Thread network and variants of Diffie-Hellman protocols, such as the STS, ISO, and SIGMA protocols [21], and the Ephemeral Diffie-Hellman Over COSE[7], and other PLS protocols.

We also plan to deploy our verified case studies on a public website, allowing researchers, engineers, or students in security to explore and verify protocols interactively using our sound animation.

Acknowledgements.

The EPSRC and DSIT support this work through the Communications Hub for Empowering Distributed Cloud Computing Applications and Research (CHEDDAR) under grants EP/X040518/1, EP/Y037421/1 and EP/Y019229/1. We thank Michele Sevegnani, Yue Gu, and Sana Hafeez from the University of Glasgow, Dalal Alrajeh, and Zhi Zhang from Imperial College London for their helpful comments on an early version of the manuscript.

[7] https://datatracker.ietf.org/doc/rfc9528/.

References

1. Arapinis, M., Mancini, L., Ritter, E., Ryan, M., Golde, N., Redon, K., Borgaonkar, R.: New privacy issues in mobile telephony: fix and verification. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. p. 205–216. CCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2382196.2382221, https://doi.org/10.1145/2382196.2382221
2. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications, pp. 281–285. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/11513988_27
3. Basin, D., Cremers, C., Dreier, J., Sasse, R.: Symbolically analyzing security protocols using tamarin. ACM SIGLOG News **4**(4), 19–30 (Nov 2017). https://doi.org/10.1145/3157831.3157835
4. Basin, D., Cremers, C., Dreier, J., Sasse, R.: Tamarin: Verification of Large-Scale, Real World, Cryptographic Protocols. IEEE Security and Privacy Magazine (2022). https://doi.org/10.1109/msec.2022.3154689, https://hal.science/hal-03586826
5. Basin, D., Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R., Stettler, V.: A Formal Analysis of 5G Authentication. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 1383–1396. CCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243846, https://doi.org/10.1145/3243734.3243846
6. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE (May 2017). https://doi.org/10.1109/sp.2017.26
7. Blake-Wilson, S., Menezes, A.: Authenticated Diffe-Hellman Key Agreement Protocols, pp. 339–361. Springer Berlin Heidelberg (1999). https://doi.org/10.1007/3-540-48892-8_26
8. Blanchet, B.: Modeling and verifying security protocols with the applied pi calculus and proverif. Foundations and Trends® in Privacy and Security **1**(1–2), 1–135 (2016). https://doi.org/10.1561/3300000004
9. Boichut, Y., Genet, T., Glouche, Y., Heen, O.: Using animation to improve formal specifications of security protocols. In: 2nd Conference on Security in Network Architectures and Information Systems (SARSSI 2007). pp. 169–182 (2007)
10. Costa, G., Degano, P., Galletta, L., Soderi, S.: Formally verifying security protocols built on watermarking and jamming. Computers & Security **128**, 103133 (May 2023). https://doi.org/10.1016/j.cose.2023.103133, https://www.sciencedirect.com/science/article/pii/S0167404823000433
11. Cox, I., Miller, M., McKellips, A.: Watermarking as communications with side information. Proceedings of the IEEE **87**(7), 1127–1141 (Jul 1999). https://doi.org/10.1109/5.771068, https://ieeexplore.ieee.org/document/771068
12. Csiszár, I.: Almost Independence and Secrecy Capacity. Probl. Peredachi Inf., **32:1**, 48–57 (1996)
13. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638

14. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2), 198–208 (1983). https://doi.org/10.1109/TIT.1983.1056650
15. Escobar, S., Meadows, C., Meseguer, J.: A rewriting-based inference system for the NRL Protocol Analyzer and its meta-logical properties. Theoretical Computer Science **367**(1–2), 162–202 (Nov 2006). https://doi.org/10.1016/j.tcs.2006.08.035
16. Foster, S., Hur, C.K., Woodcock, J.: Formally Verified Simulations of State-Rich Processes Using Interaction Trees in Isabelle/HOL. In: Haddad, S., Varacca, D. (eds.) 32nd International Conference on Concurrency Theory (CONCUR 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 203, pp. 20:1–20:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.CONCUR.2021.20, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2021.20
17. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.W.: FDR3 — A Modern Refinement Checker for CSP. In: Ábrahám, E., Havelund, K. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 187–201. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_13
18. Gollakota, S., Katabi, D.: Physical layer wireless security made fast and channel independent. In: 2011 Proceedings IEEE INFOCOM. pp. 1125–1133 (Apr 2011). https://doi.org/10.1109/INFCOM.2011.5934889, https://ieeexplore.ieee.org/document/5934889
19. Haftmann, F., Nipkow, T.: Code Generation via Higher-Order Rewrite Systems. In: Blume, M., Kobayashi, N., Vidal, G. (eds.) Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6009, pp. 103–117. Springer (2010). https://doi.org/10.1007/978-3-642-12251-4_9, https://doi.org/10.1007/978-3-642-12251-4_9
20. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall Int. (1985)
21. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 400–425. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_24
22. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol: (Extended Abstract), pp. 546–566. Springer Berlin Heidelberg (2005). https://doi.org/10.1007/11535218_33
23. Kumar, Megha.S., Ramanathan, R., Jayakumar, M.: Key less physical layer security for wireless networks: A survey. Engineering Science and Technology, an International Journal **35**, 101260 (Nov 2022). https://doi.org/10.1016/j.jestch.2022.101260, https://www.sciencedirect.com/science/article/pii/S2215098622001690
24. Li, X., Yu, C., Hizlan, M., Kim, W.T., Park, S.: Physical Layer Watermarking of Direct Sequence Spread Spectrum Signals. In: MILCOM 2013 - 2013 IEEE Military Communications Conference. pp. 476–481. IEEE, San Diego, CA, USA (Nov 2013). https://doi.org/10.1109/MILCOM.2013.88, http://ieeexplore.ieee.org/document/6735668/
25. Lowe, G.: Casper: A Compiler for the Analysis of Security Protocols. In: Proceedings 10th Computer Security Foundations Workshop. pp. 18–30 (Jun 1997). https://doi.org/10.1109/CSFW.1997.596779, https://ieeexplore.ieee.org/document/596779

26. Lowe, G.: Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR, pp. 147–166. Springer Berlin Heidelberg (1996). https://doi.org/10.1007/3-540-61042-1_43
27. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., USA, 1st edn. (1996). https://doi.org/10.1201/9780429466335
28. Metere, R., Arnaboldi, L.: Automating cryptographic protocol language generation from structured specifications. In: Proceedings of the IEEE/ACM 10th International Conference on Formal Methods in Software Engineering. pp. 91–101 (2022). https://doi.org/10.1145/3524482.35276
29. Miller, R., Boureanu, I., Wesemeyer, S., Newton, C.J.P.: The 5G Key-Establishment Stack: In-Depth Formal Verification and Experimentation. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. p. 237–251. ASIA CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3488932.3517421, https://doi.org/10.1145/3488932.3517421
30. Mitev, M., Chorti, A., Poor, H.V., Fettweis, G.P.: What Physical Layer Security Can Do for 6G Security. IEEE Open Journal of Vehicular Technology **4**, 375–388 (2023). https://doi.org/10.1109/OJVT.2023.3245071, https://ieeexplore.ieee.org/document/10044975/?arnumber=10044975
31. Miyazawa, A., Ribeiro, P., Li, W., Cavalcanti, A., Timmis, J., Woodcock, J.: Robochart: modelling and verification of the functional behaviour of robotic applications. Software & Systems Modeling **18**(5), 3097–3149 (Jan 2019). https://doi.org/10.1007/s10270-018-00710-z
32. Needham, R.M., Schroeder, M.D.: Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM **21**(12), 993–999 (Dec 1978). https://doi.org/10.1145/359657.359659, https://dl.acm.org/doi/10.1145/359657.359659
33. Nguyen, V.L., Lin, P.C., Cheng, B.C., Hwang, R.H., Lin, Y.D.: Security and Privacy for 6G: A Survey on Prospective Technologies and Challenges. IEEE Communications Surveys & Tutorials **23**(4), 2384–2428 (2021). https://doi.org/10.1109/COMST.2021.3108618
34. Nipkow, T., Wenzel, M., Paulson, L.C.: Isabelle/HOL: a proof assistant for higher-order logic. Springer (2002). https://doi.org/10.1007/3-540-45949-9
35. Paulson, L.C.: Mechanized proofs of security protocols: Needham-schroeder with public keys. Tech. rep., University of Cambridge (1997). https://doi.org/10.48456/tr-413
36. Roscoe, A.W.: Understanding Concurrent Systems. Texts in Computer Science, Springer (2011). https://doi.org/10.1007/978-1-84882-258-0
37. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (Jul 1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x, https://ieeexplore.ieee.org/document/6773024/?arnumber=6773024
38. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (1994). https://doi.org/10.1109/SFCS.1994.365700
39. Soderi, S., Brighente, A., Turrin, F., Conti, M.: VLC Physical Layer Security through RIS-aided Jamming Receiver for 6G Wireless Networks. In: 2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). pp. 370–378 (Sep 2022). https://doi.org/10.1109/SECON55815.2022.9918547, http://arxiv.org/abs/2205.09026

40. Soderi, S., De Nicola, R.: 6G Networks Physical Layer Security Using RGB Visible Light Communications. IEEE Access **10**, 5482–5496 (2022). https://doi.org/10.1109/ACCESS.2021.3139456
41. Soderi, S., Mucchi, L., Hämäläinen, M., Piva, A., Iinatti, J.: Physical layer security based on spread-spectrum watermarking and jamming receiver. Transactions on Emerging Telecommunications Technologies **28**(7), e3142 (2017). https://doi.org/10.1002/ett.3142, https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3142
42. Wyner, A.D.: The wire-tap channel. The Bell System Technical Journal **54**(8), 1355–1387 (Oct 1975). https://doi.org/10.1002/j.1538-7305.1975.tb02040.x, https://ieeexplore.ieee.org/document/6772207
43. Xia, L.y.: Executable Denotational Semantics with Interaction Trees. Ph.D. thesis, University of Pennsylvania (2022)
44. Xia, L.y., Zakowski, Y., He, P., Hur, C.K., Malecha, G., Pierce, B.C., Zdancewic, S.: Interaction Trees: Representing Recursive and Impure Programs in Coq. Proc. ACM Program. Lang. **4**(POPL) (Dec 2019). https://doi.org/10.1145/3371119, https://doi.org/10.1145/3371119
45. Ye, K., Cavalcanti, A., Foster, S., Miyazawa, A., Woodcock, J.: Probabilistic modelling and verification using RoboChart and PRISM. Softw. Syst. Model. **21**(2), 667–716 (2022). https://doi.org/10.1007/s10270-021-00916-8, https://doi.org/10.1007/s10270-021-00916-8
46. Ye, K., Foster, S., Woodcock, J.: Formally verified animation for RoboChart using interaction trees. Journal of Logical and Algebraic Methods in Programming **137**, 100940 (Feb 2024). https://doi.org/10.1016/j.jlamp.2023.100940
47. Ye, K., Metere, R., Yadav, P.: User-Guided Verification of Security Protocols via Sound Animation. In: Software Engineering and Formal Methods. pp. 33–51. Springer Nature Switzerland (Nov 2024). https://doi.org/10.1007/978-3-031-77382-2_3
48. Zhang, S., Zhu, D., Liu, Y.: Artificial intelligence empowered physical layer security for 6G: State-of-the-art, challenges, and opportunities. Computer Networks **242**, 110255 (Apr 2024). https://doi.org/10.1016/j.comnet.2024.110255, https://www.sciencedirect.com/science/article/pii/S1389128624000872

A Soundness illustration

Figure 4 shows how the definition of a CSP operator `outp` in ITrees, on the left of the diagram, is linked to the final Haskell function `outp` on the right of the diagram. This link is automated in Isabelle/HOL.

B Message type

The message type in our framework is defined below.

```
datatype ('a, 'n, 'k, 's, 'g, 'bm, 'bl::len) dmsg =
  MAg (ma:"'a dagent")
| MNon (mn:"'n dnonce")
| MK (mk:"('k, 's) dkey")
| MPair (mc1:"T dmsg") (mc2:"T dmsg")
```

⟨_,_⟩

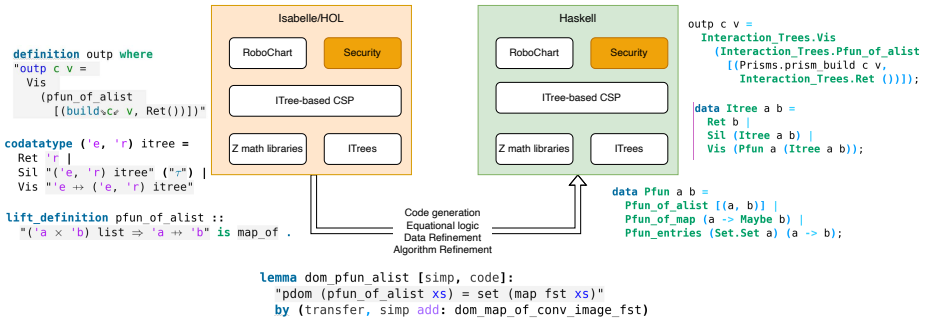


Fig. 4. An example to illustrate how a CSP operator `outp` (output a value `v` on channel `c`) is implemented on ITrees using associative lists (`alist`) to represent partial functions (`pfun`), and their counterparts in generated Haskell code where `dom_pfun_alist` is a proved lemma in Isabelle/HOL and based on equational logic to replace its left-hand side by its right-hand side in code generation to preserve semantics. We say our approach from CSP specification to the resultant Haskell code forms an unbroken link.

`| MExpg (eg:"'g dexpg")`	$g_m\_$
`| MModExp (mmem:"T dmsg") (mmek:"T dmsg")`	$\_^{\frown}\_$ *or* $\_\hat{}\_$
`| MBitm (mbm:"('bm, 'bl) dbitmask")`	$b_m\_\ \_$
`| MWat (mwm:"T dmsg") (mwb:"T dmsg")`	$\{\!\|\_\|\!\}^w_\_$
`| MJam (mjm:"T dmsg") (mjb:"T dmsg")`	$\{\!\|\_\|\!\}^j_\_$
`| MSEnc (msem:"T dmsg") (msek:"T dmsg")`	$\{\!\|\_\|\!\}^s_\_$
`| MAEnc (mem:"T dmsg") (mek:"T dmsg")`	$\{\!\|\_\|\!\}^a_\_$
`| MSig (msd:"T dmsg") (msk:"T dmsg")`	$\{\!\|\_\|\!\}^d_\_$

C Web interface

We illustrate our web interface in Figure 5 and Figure 6a.

Figure 5 shows the manual exploration interface for NSWJ3, where the upper part of the table displays current enabled and available events for users to choose for animation. These events are numbered from 1. In the figure, there are three enabling events from 1 to 3. Each event has a unique identity and is shown in the second column. Additionally, the channel name, source, medium, target agents, and message body of an event are displayed. While Env is not a defined agent in our protocol, we use it to denote the environment of protocols or the users within them. For example, the second event is a message watermarked by Alice's bitmask (BM0) from a paired message composed of the nonce (N0) and the identity (A0) of Alice, sent by Intruder and received by Bob over channel `Recv`.

In the middle of the diagram, three functions are provided: (1) choose one event (1 to 3 in this case) from all enabled events to animate; (2) reset manual animation so users can start from the beginning again; and (3) choose the location of Eve for animation. When users perform the animation, their interaction history will be dynamically displayed as a sequence diagram at the bottom. It is worth

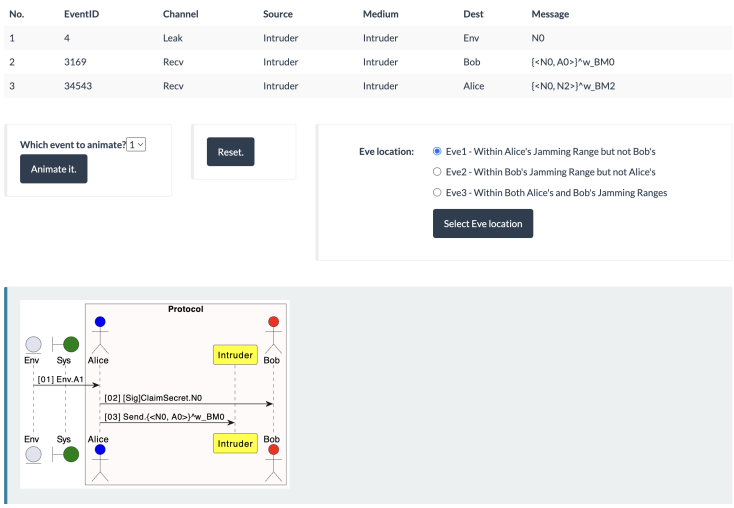
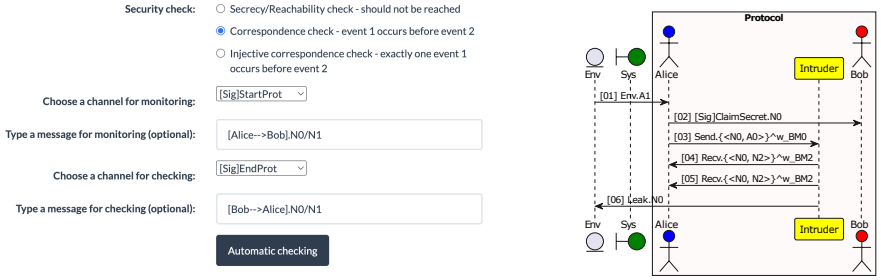


Fig. 5. Manual animation: choose one numbered event to animate, and the interaction history is dynamically displayed in the sequence diagram.

mentioning that the web interface supports multiple sessions, allowing users to animate the same protocol or different protocols simultaneously.



(a) Configuration interface (b) Secrecy counterexample

Fig. 6. Automatic reachability and correspondence checking for secrecy and authenticity.

Figure 6a shows the automatic checking interface, where users can configure what property to check and its specific input. Now, three kinds of properties can be verified: secrecy, correspondence, and injective correspondence. For secrecy, users need to choose a channel for checking (such as Leak, Terminate, or Signal) and then optionally input the message in the input box. If no message is specified, the tool will check the reachability of all events on the chosen channel. For correspondence, users need to configure the event (event 1) for monitoring in addition

to the event (event 2) for reachability checking. For example, the figure shows a configuration to check the authenticity of Bob in NSWJ3: check the reachability of the signal `EndProt` for Bob to finish the run with Alice using their nonces N0 and N1 while monitoring the occurrence of the signal `StartProt` for Alice to start the run with Bob using the same nonces. If the `EndProt` is reached without the occurrence of the signal `StartProt` before it, a counterexample is displayed. Then, users can click a button to view the counterexample in a sequence diagram.

Figure 6b shows a counterexample for the secrecy checking of NSWJ when Eve1 is chosen. Alice's nonce (N0) is leaked because Eve is located outside Bob's jamming range. When Alice sends a watermarked message, Intruder can derive its plain message using the inference rule Wat1 in Table 2 and learn N0. Then Intruder leaks it into the environment. This is not surprising because the intruder can hear watermarked messages from Alice or Bob. So, based on the watermarking rule in Table 2, the clear message is derived. What is more surprising is authenticity, which holds even when secrecy is violated. The original NSPK protocol is insecure, and both properties are violated [26,47]. The clear explanation is that watermarking prevents authenticity from being compromised; in particular, though the intruder can learn the clear message, they cannot fake a watermarked message using Alice's or Bob's watermarking bitmask because they do not know these bitmasks.

Both our terminal and web interfaces allow users to do user-guided verification. For the web interface, users can manually explore a protocol for some steps and then want to know if there are any security issues after this stage. They can utilise the automatic verification function to conduct checks. In this case, the verification will start from the explored state instead of from the beginning. This could reduce the complexity of verifying large protocols and make them more scalable.