A Unified Symbolic Analysis of WireGuard

Pascal Lafourcade[1, 2] Dhekra Mahmoud[1,2] Sylvain Ruhault[3]

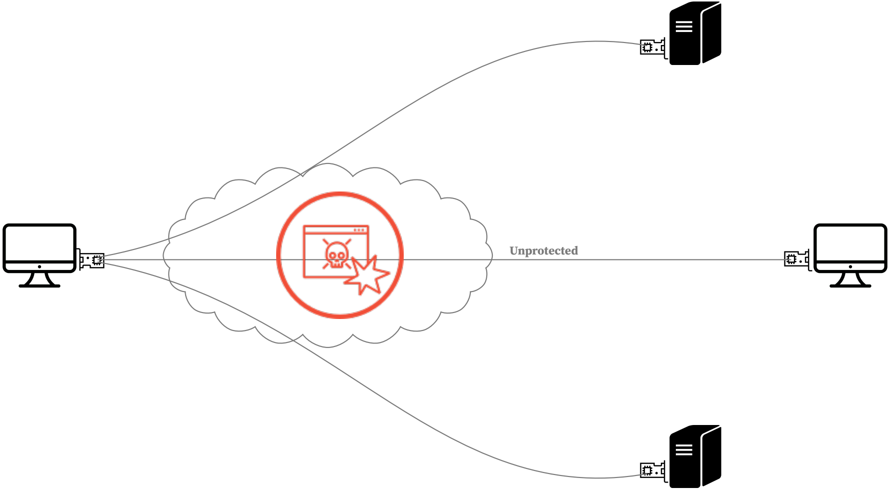
[1] Université Clermont Auvergne,

[2] Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes,

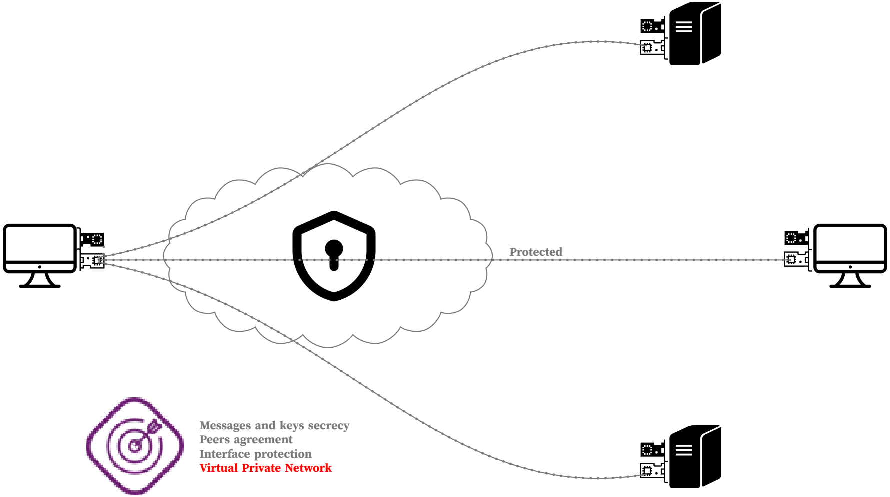
[3] Agence Nationale de la Sécurité des Systèmes d'Information

April 11, 2024

Context - VPN

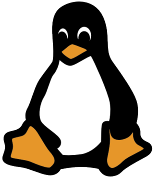


Context - VPN



Messages and keys secrecy
Peers agreement
Interface protection
Virtual Private Network

WIREGUARD
FAST, MODERN, SECURE VPN TUNNEL



WIREGUARD
FAST, MODERN, SECURE VPN TUNNEL

What about **Privacy**?

Formal Verification of security protocols

Formal Verification of security protocols

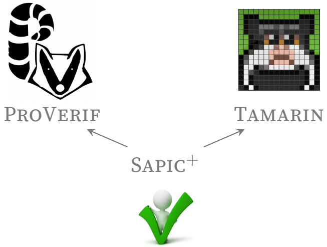
Manual proofs

- Error prone
- Tedious
- Active Adversaries
- Guarantees on security ?

Formal Verification of security protocols

Software tools

- Automated & semi-automated
- Formal proofs
- Handle protocols' complexity
- Dedicated approaches
- **Symbolic** & Computational



ProVerif

Tamarin

Sapic$^+$

Manual proofs

- Error prone
- Tedious
- Active Adversaries
- Guarantees on security ?

Current symbolic analyses

Symbolic

- 2018: J. A. Donenfeld and K. Milner, "Formal verification of the WireGuard protocol" *WireGuard*
- 2019: N. Kobeissi, G. Nicolas, and K. Bhargavan, "Noise explorer: Fully automated modeling and verification for arbitrary Noise protocols" *IKpsk2*
- 2020: G. Girol, L. Hirschi, R. Sasse, D. Jackson, C. Cremers, and D. A. Basin, "A spectral analysis of Noise: A comprehensive, automated, formal analysis of Diffie-Hellman protocols" *IKpsk2*

Threats

- Static private key reveal / set
- Ephemeral private key reveal / set
- PSK reveal / set
- Static key distribution corruption

Security Properties

- Message agreement
- Key secrecy (incl. PFS)
- Anonymity

Our target threat model for *WireGuard*

Threats

- Static private key reveal ✓ / set ✓
- Ephemeral private key reveal ✓ / set ✓
- PSK reveal ✓ / set ✓
- Static key distribution corruption ✓
- New! Pre-computation reveal ✓ / set ✓

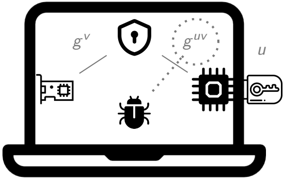
Pre-computation ?

- Static-static key :
  - Initiator $V^u = g^{uv}$
  - Responder $U^v = g^{uv}$

  before session begins, hence WireGuard maintains it.

Compromise of g^{uv} is **weaker** than compromise of u or v:

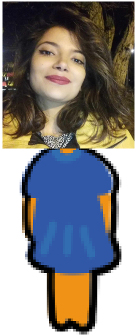
- $u \wedge g^v \implies g^{uv}$
- however $g^v \wedge g^{uv} \not\Longrightarrow u$



Results of our analysis

- Wireguard **does not** preserve users' **privacy** !
- **Necessary and Sufficient conditions** of compromise for each **security property**.

To know more about:



- Formal Verification
- Symbolic Model
- Attack on Anonymity
- And much more ...

Meet me with my Poster :-)