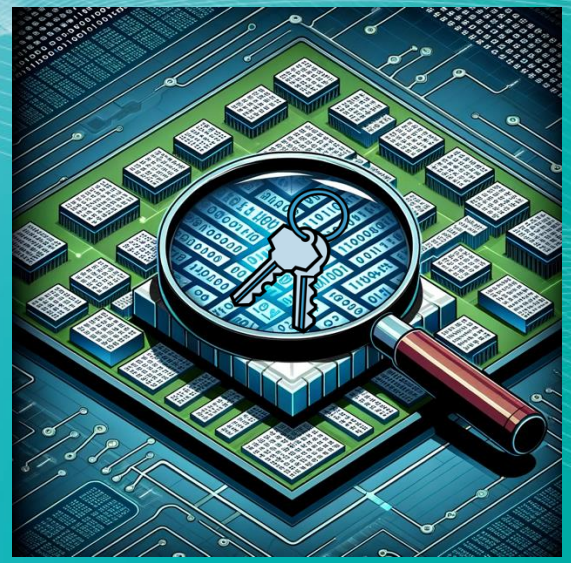
Daniel Baier*, Martin Lambertz

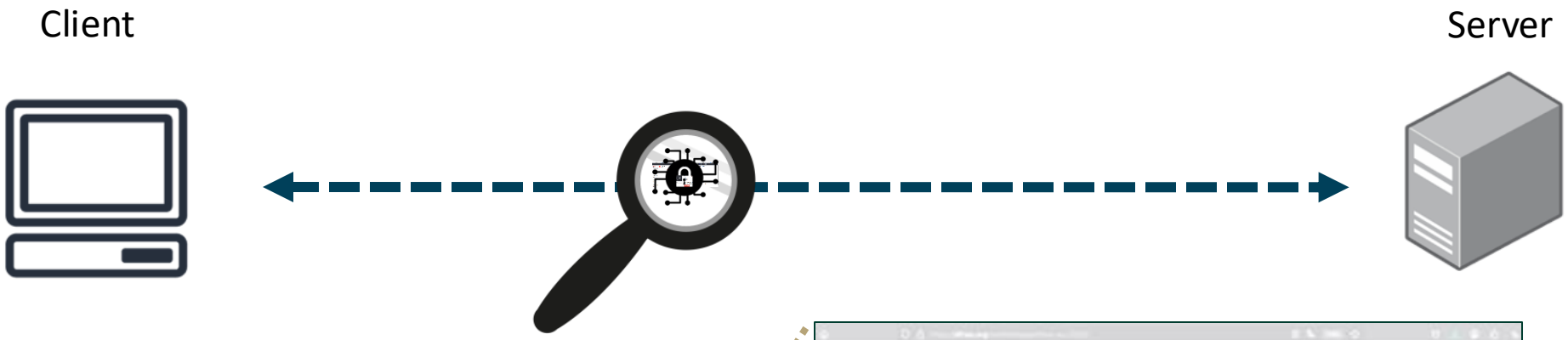
Secrets in Motion: Dynamic TLS Key Recovery

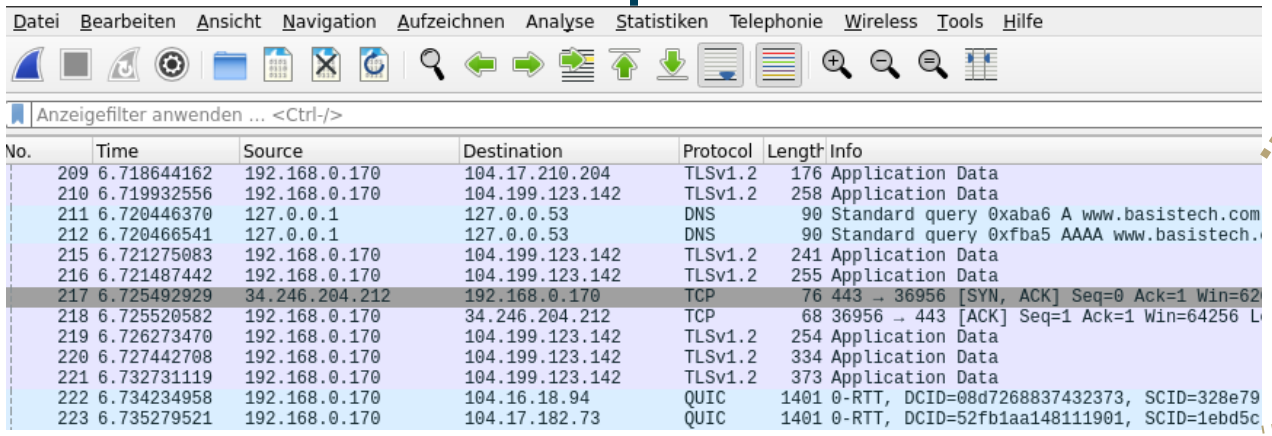
Fraunhofer FKIE

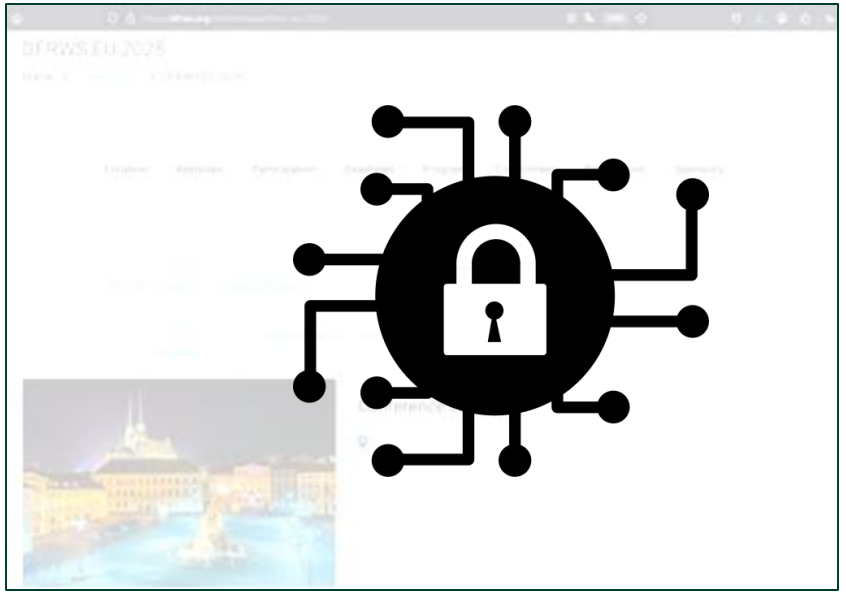
Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE



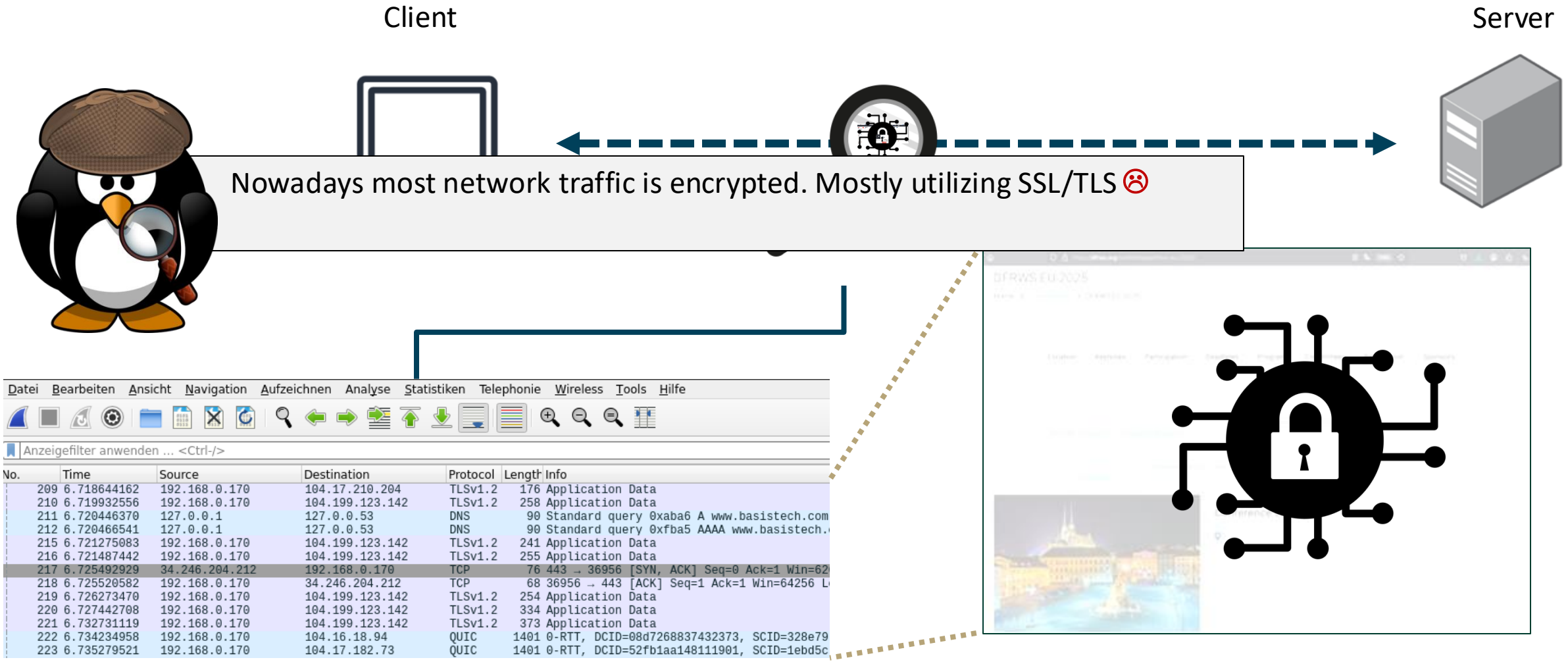
Intro
Traffic analysis today





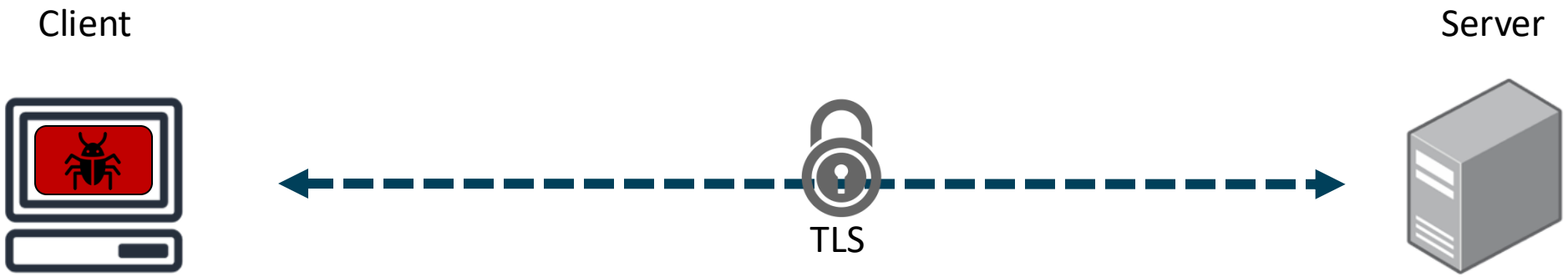


Intro
Traffic analysis today



Intro

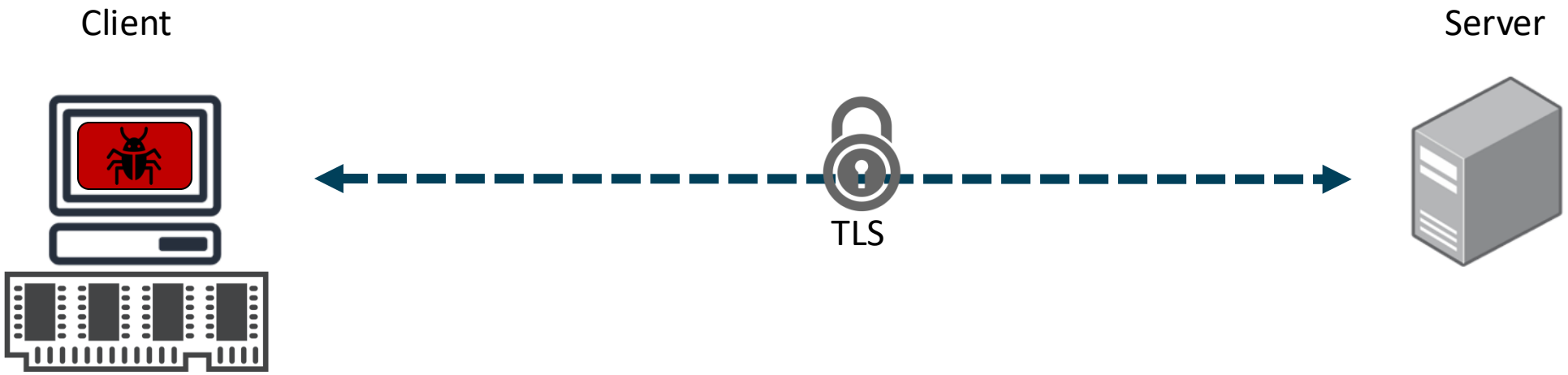
TLS encryption prevents forensic insights



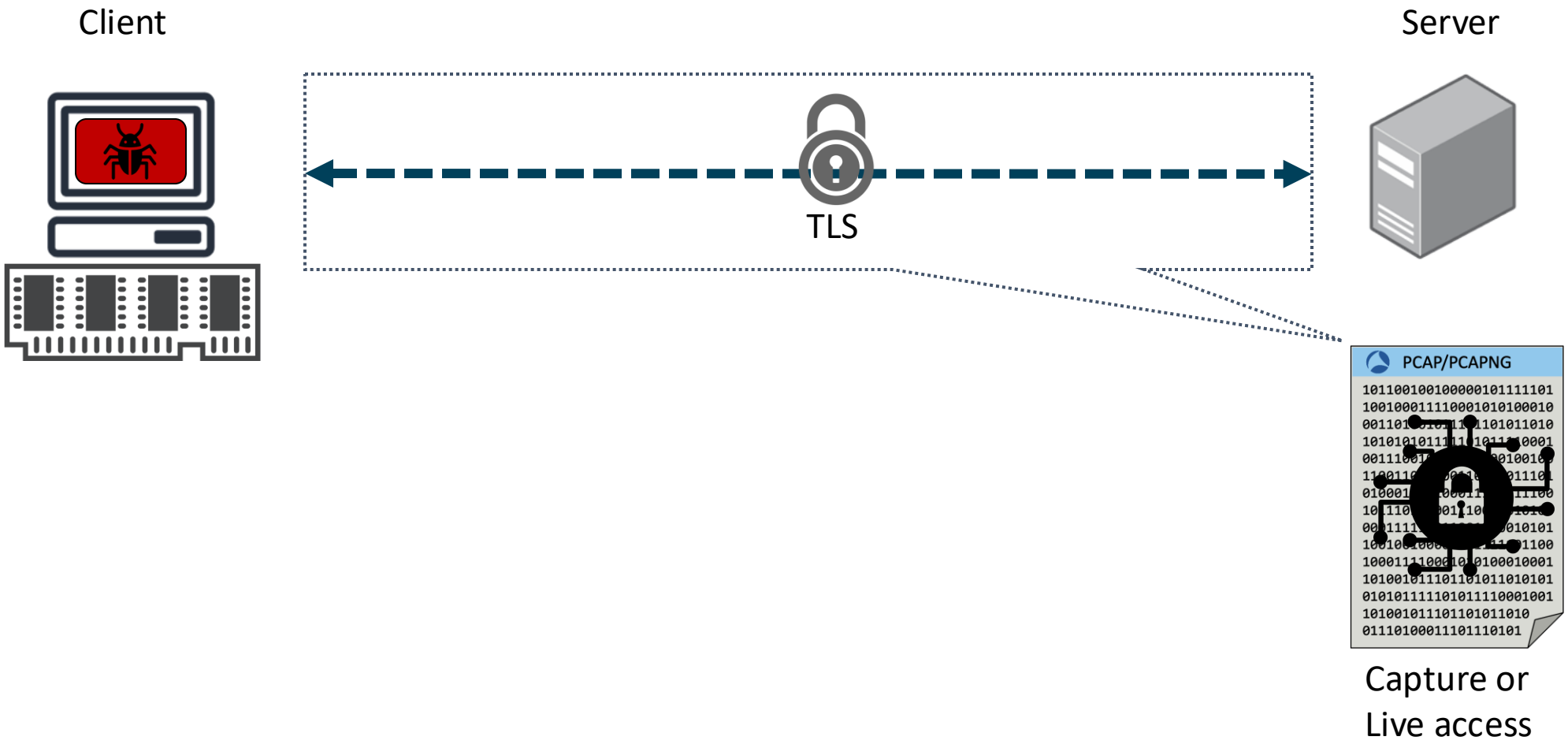
TLS encryption prevents forensic insights:

- ☒ Monitoring
- ☒ Malware analysis
- ☒ API analysis

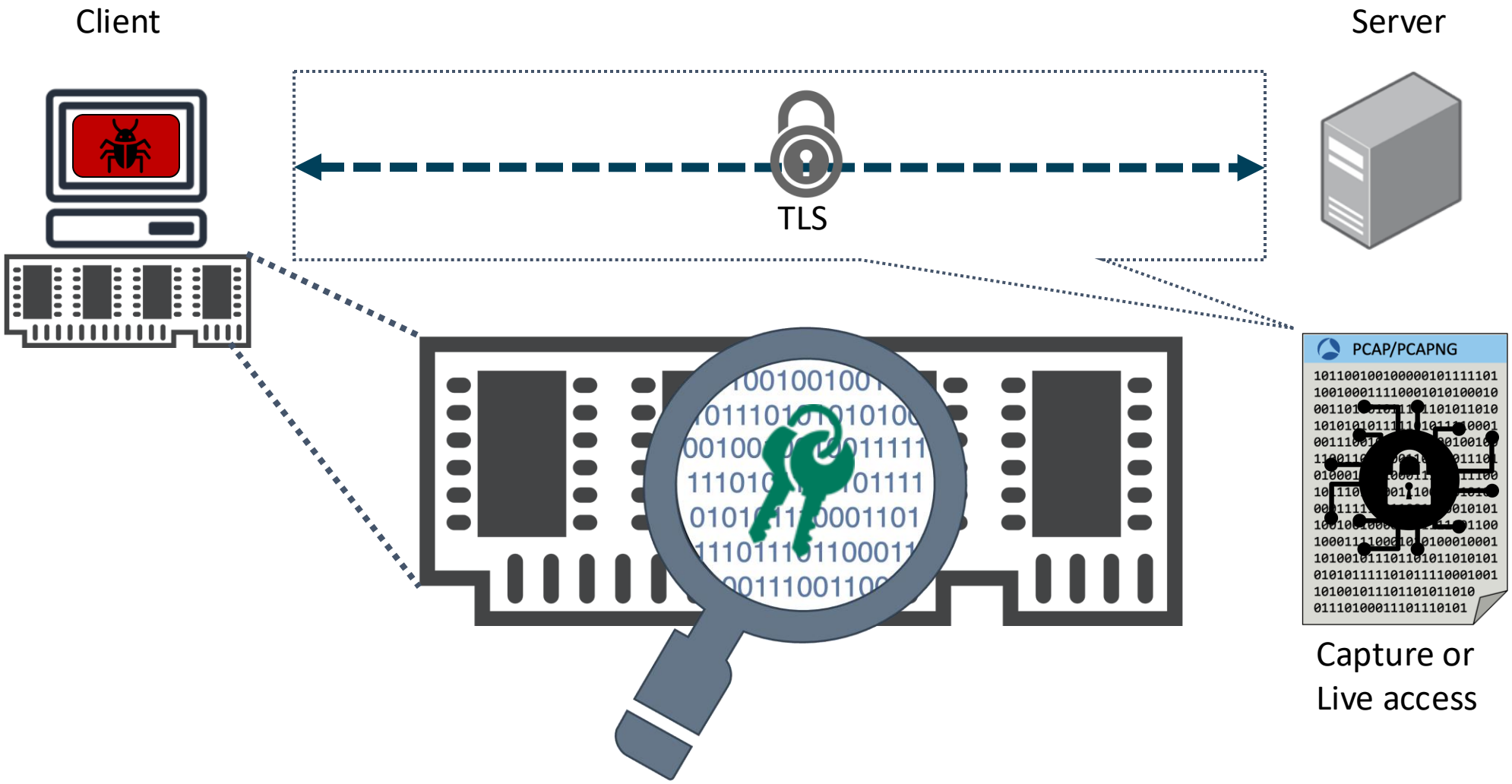
TLS Key Identification



TLS Key Identification



TLS Key Identification



Last year we systemized the current state of knowledge in the field of TLS Key Material Identification and Extraction in Memory

READ ME

*Generated by ChatGPT

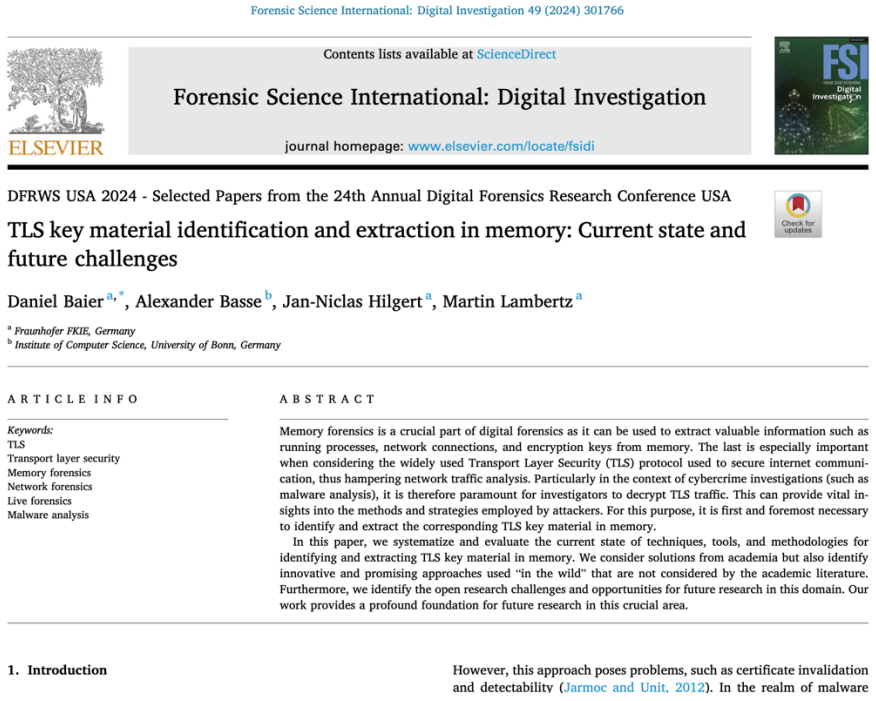
Intro

Limited TLS library support & minimal TLS 1.3 adoption

READ ME

Last year we systemized the current state of knowledge in the field of TLS Key Material Identification and Extraction in Memory

- Limited TLS library support ☹
- TLS 1.3 Key Identification and Extraction mostly unresolved…
- For more details have a look at our paper or the original ones
  - https://www.sciencedirect.com/science/article/pii/S2666281724000854



Forensic Science International: Digital Investigation 49 (2024) 301766

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA

TLS key material identification and extraction in memory: Current state and future challenges

Daniel Baier [a,*], Alexander Basse [b], Jan-Niclas Hilgert [a], Martin Lambertz [a]

[a] Fraunhofer FKIE, Germany
[b] Institute of Computer Science, University of Bonn, Germany

ARTICLE INFO

Keywords:
TLS
Transport layer security
Memory forensics
Network forensics
Live forensics
Malware analysis

ABSTRACT

Memory forensics is a crucial part of digital forensics as it can be used to extract valuable information such as running processes, network connections, and encryption keys from memory. The last is especially important when considering the widely used Transport Layer Security (TLS) protocol used to secure internet communication, thus hampering network traffic analysis. Particularly in the context of cybercrime investigations (such as malware analysis), it is therefore paramount for investigators to decrypt TLS traffic. This can provide vital insights into the methods and strategies employed by attackers. For this purpose, it is first and foremost necessary to identify and extract the corresponding TLS key material in memory.

In this paper, we systematize and evaluate the current state of techniques, tools, and methodologies for identifying and extracting TLS key material in memory. We consider solutions from academia but also identify innovative and promising approaches used "in the wild" that are not considered by the academic literature. Furthermore, we identify the open research challenges and opportunities for future research in this domain. Our work provides a profound foundation for future research in this crucial area.

1. Introduction

However, this approach poses problems, such as certificate invalidation and detectability (Jarmoc and Unit, 2012). In the realm of malware

Intro
Limited TLS library support & minimal TLS 1.3 adoption

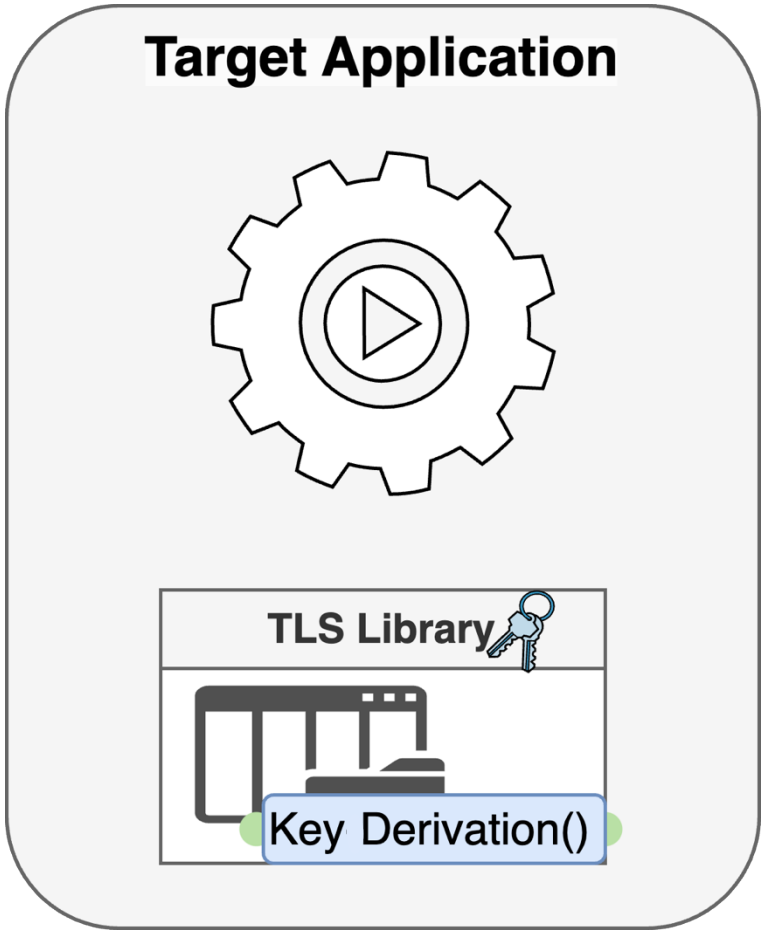
We are unable to extract the TLS key materials





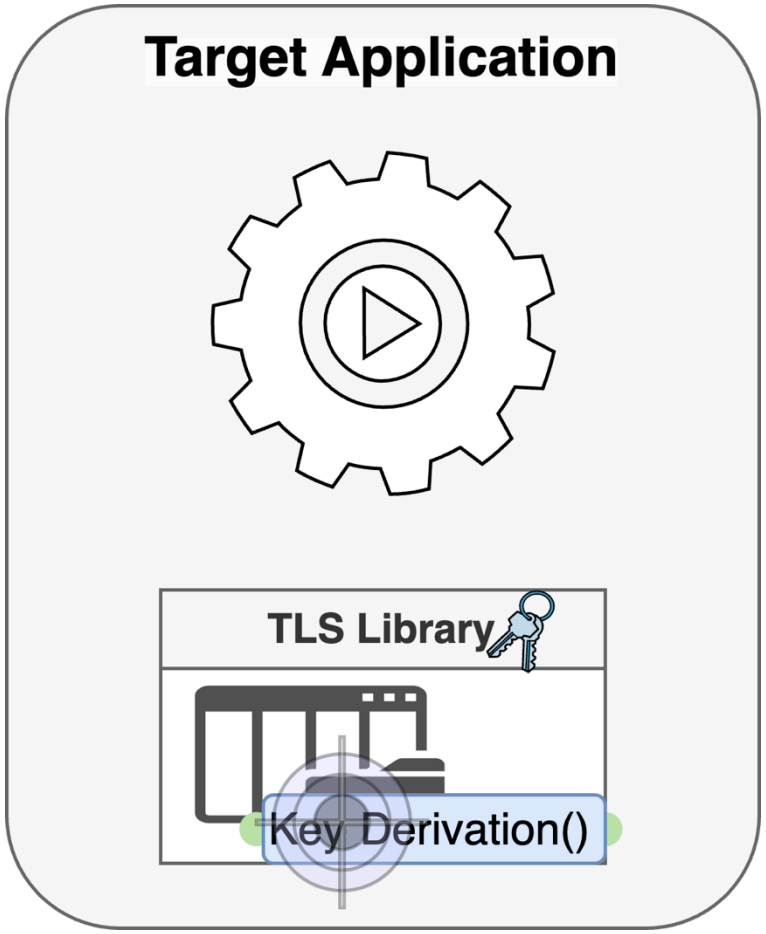
What’s the Best Way to Extract TLS Keys from Live Memory?
dfrws.org
*Generated by ChatGPT

Intercepting the Key Derivation Function
Idea



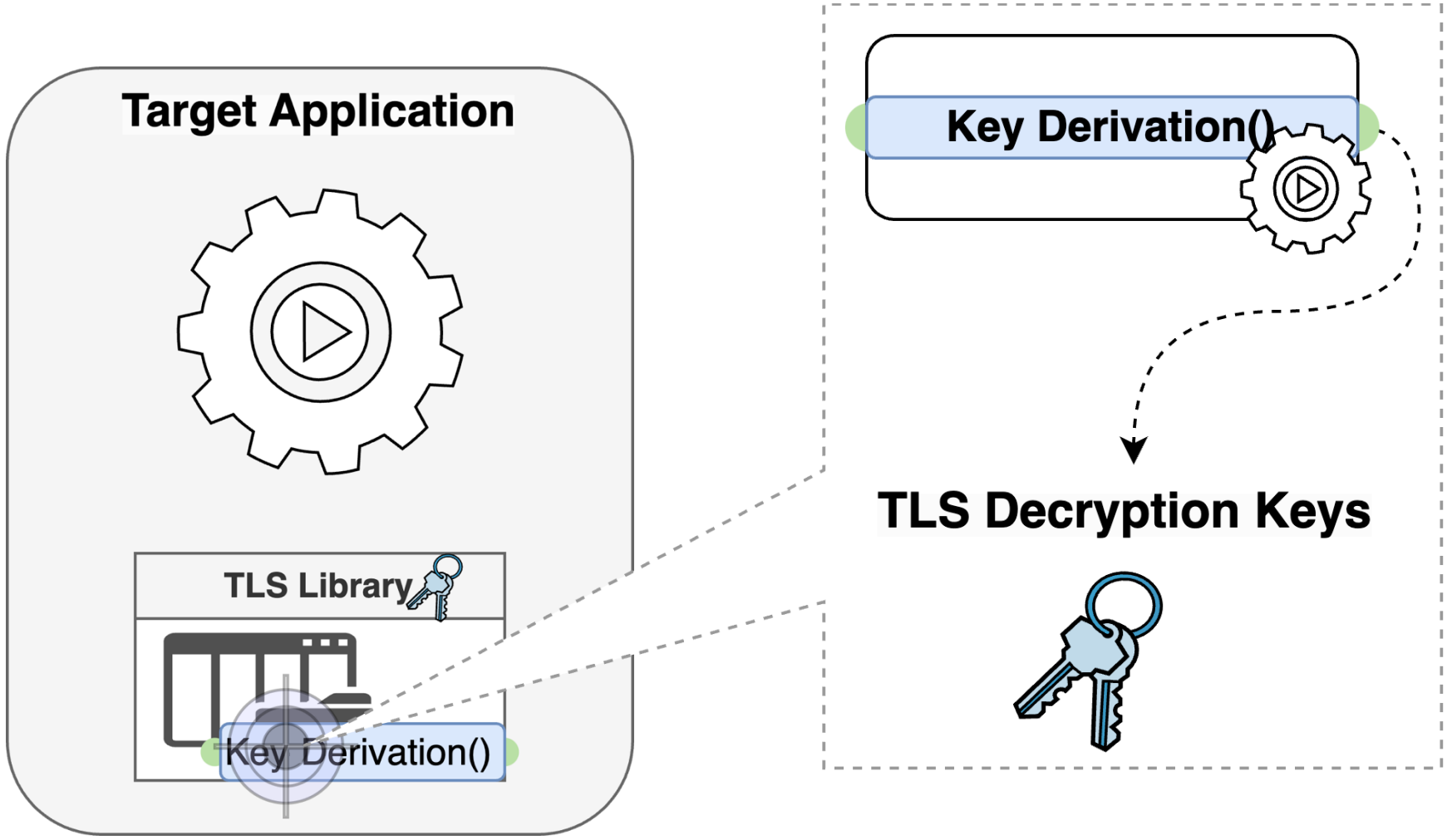
Intercepting the Key Derivation Function

Idea



Intercepting the Key Derivation Function

Idea



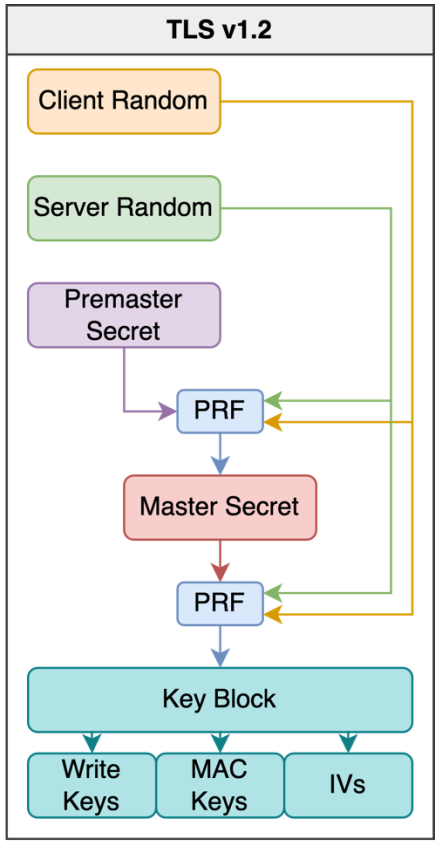
Understanding the TLS Key Schedule

*Generated by ChatGPT

Secrets in Motion: Dynamic TLS Key Recovery

Fundamentals

Decryption Keys ≤ TLS 1.2

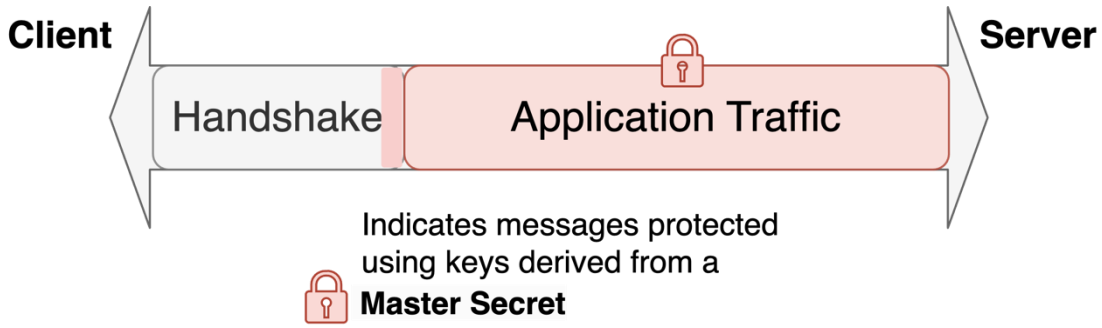


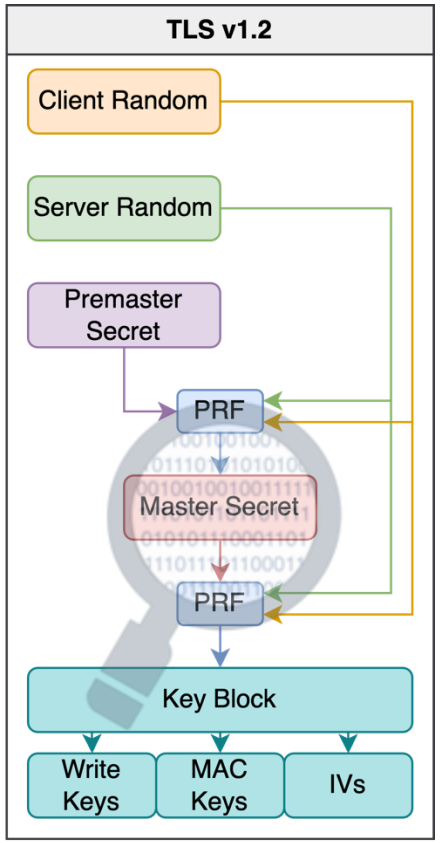
≙ Key Derivation Function

Secrets in Motion: Dynamic TLS Key Recovery

Fundamentals

Decryption Keys ≤ TLS 1.2



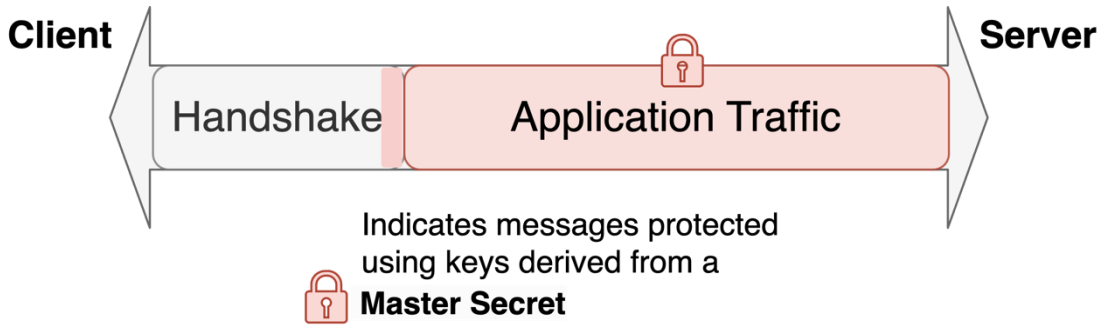


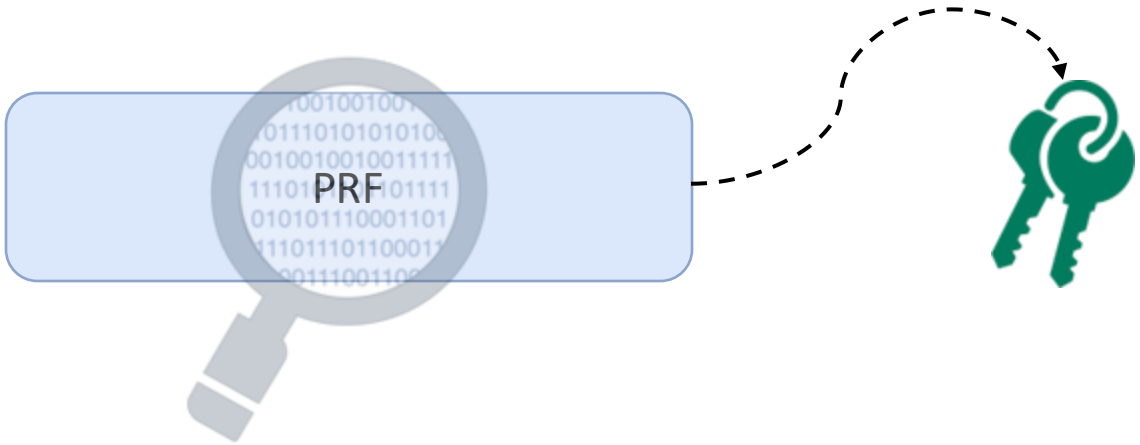
≙ Key Derivation Function

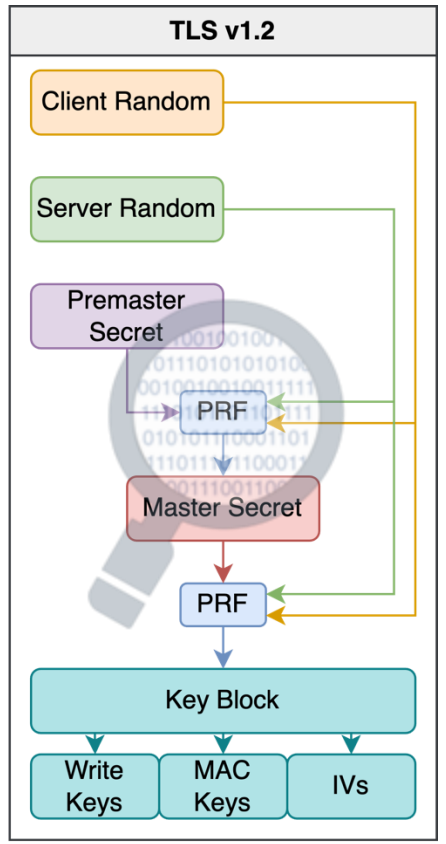
Secrets in Motion: Dynamic TLS Key Recovery

Fundamentals

Decryption Keys ≤ TLS 1.2





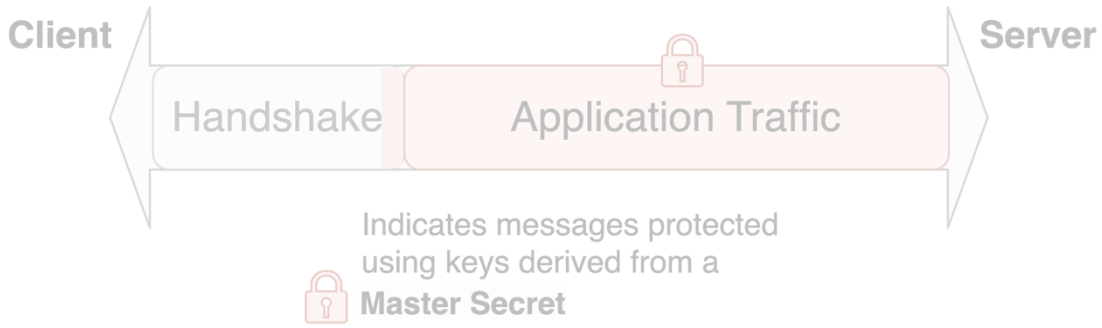


≙ Key Derivation Function

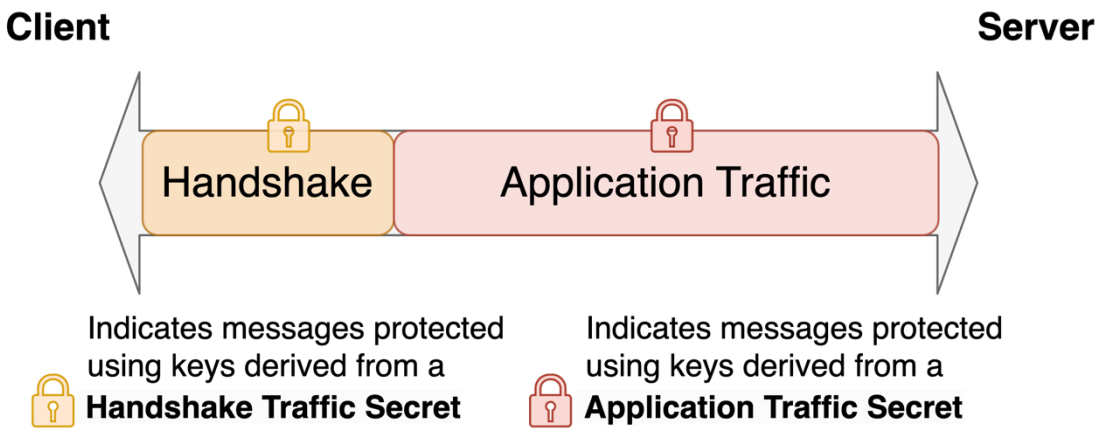
Secrets in Motion: Dynamic TLS Key Recovery

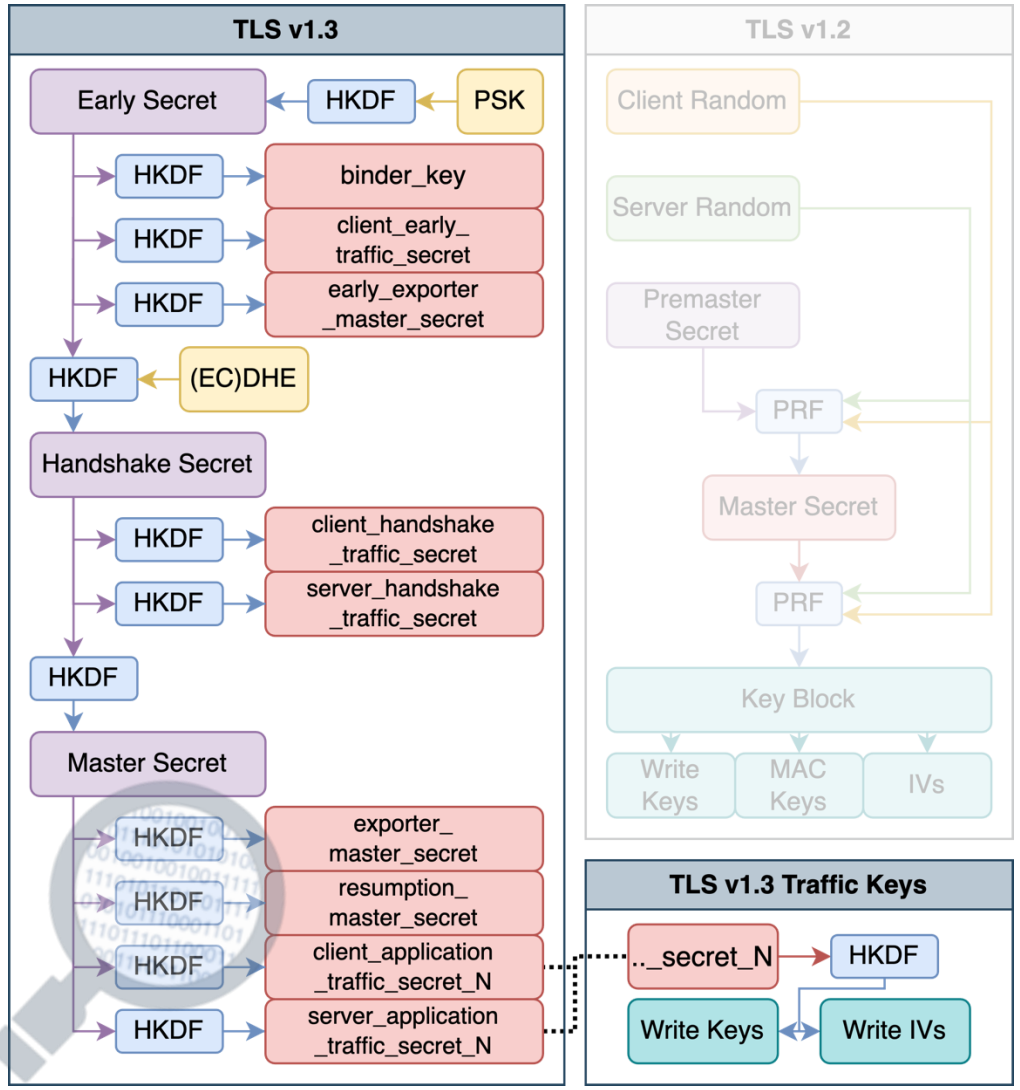
Fundamentals

Decryption Keys ≤ TLS 1.2



Decryption Keys TLS 1.3

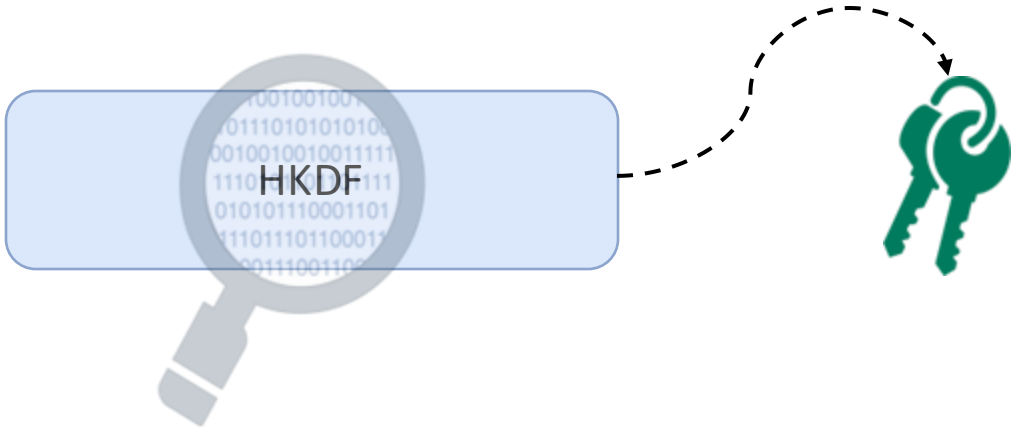




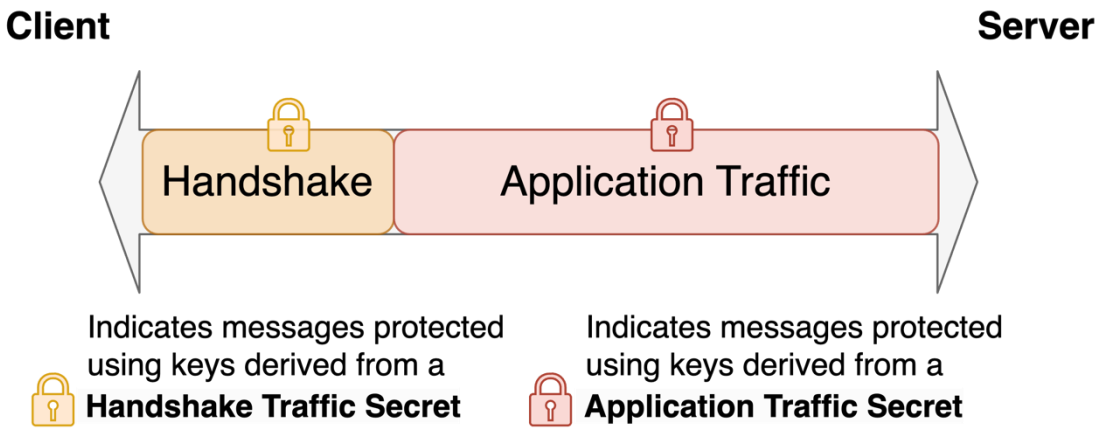
Key Derivation Function

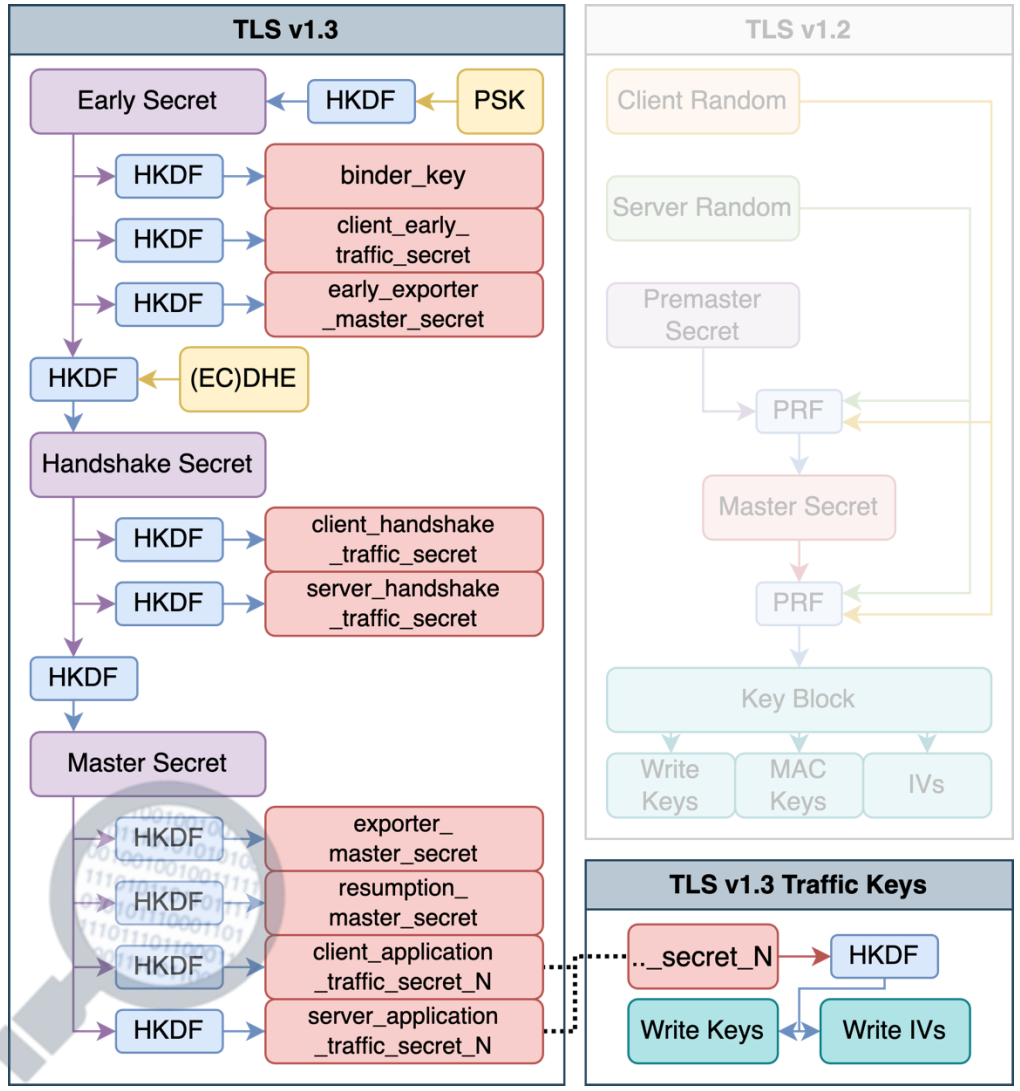
Secrets in Motion: Dynamic TLS Key Recovery

Fundamentals



Decryption Keys TLS 1.3

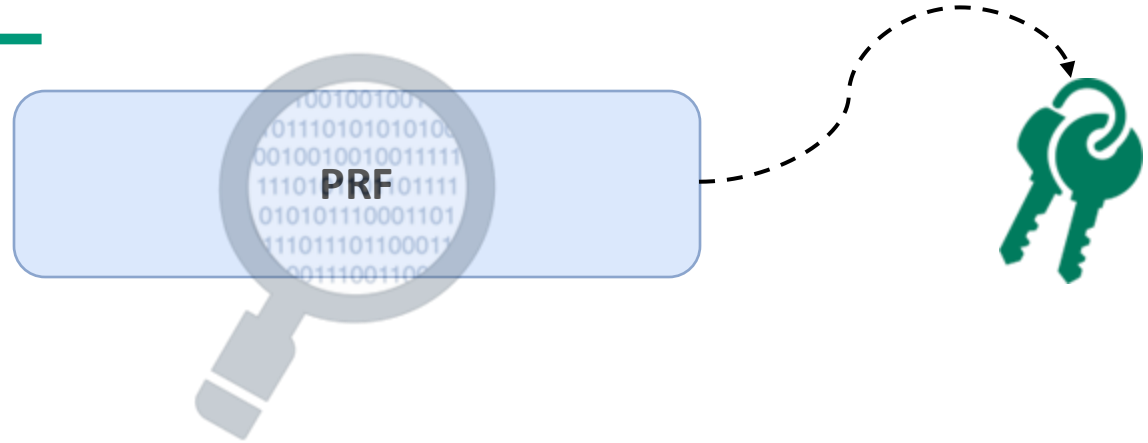




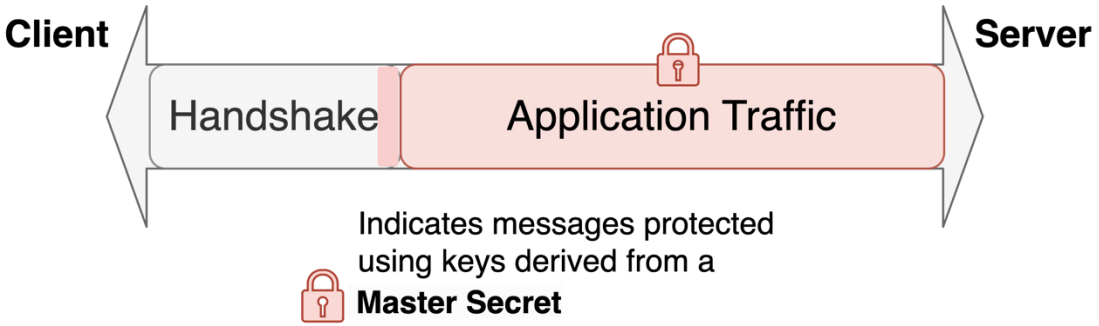
Key Derivation Function

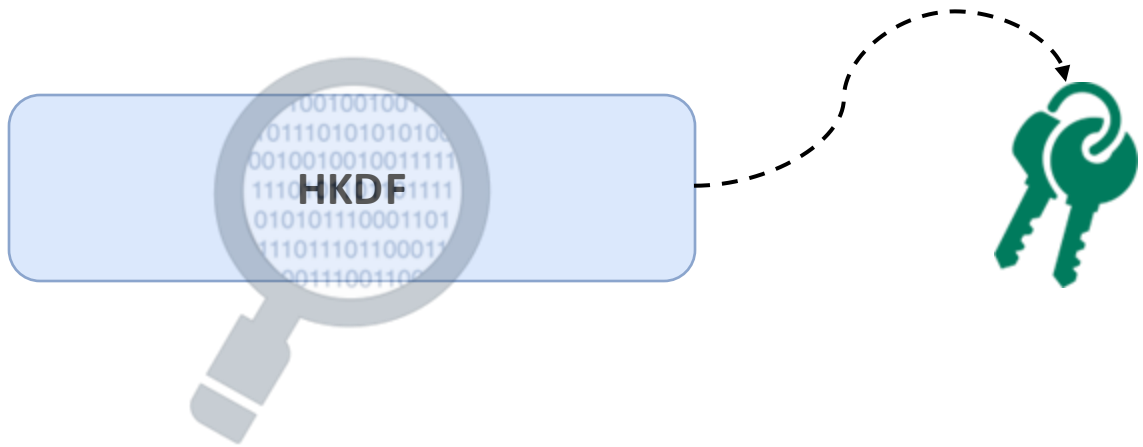
Secrets in Motion: Dynamic TLS Key Recovery

Fundamentals

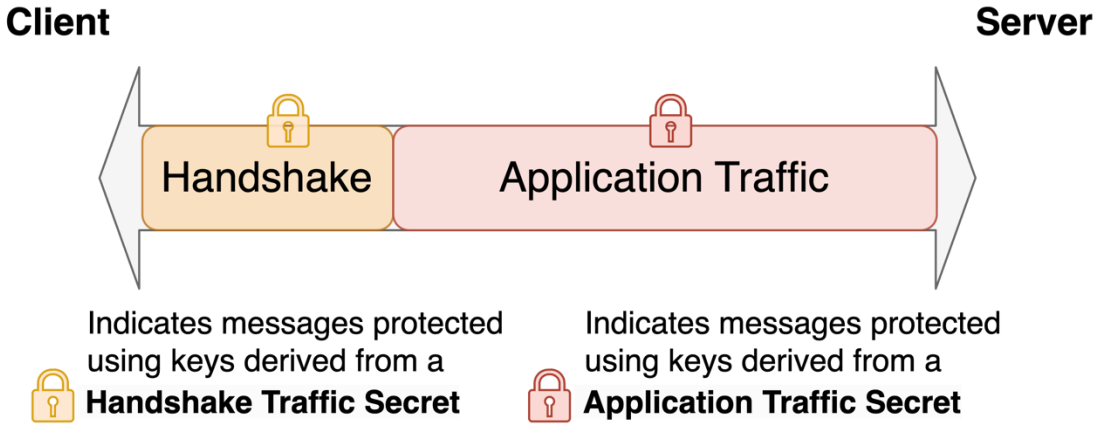


Decryption Keys ≤ TLS 1.2

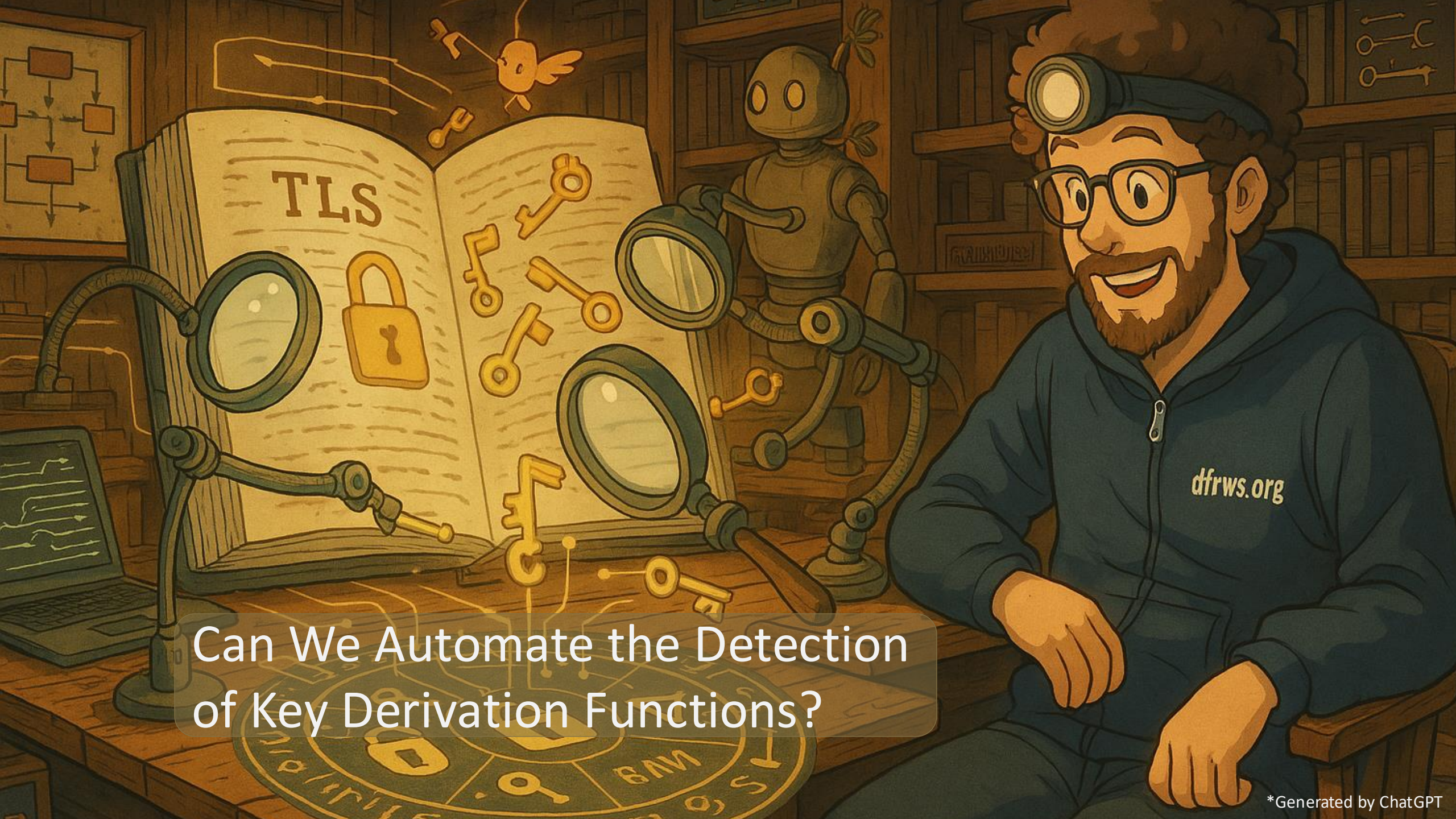




Decryption Keys TLS 1.3



Key Derivation Function



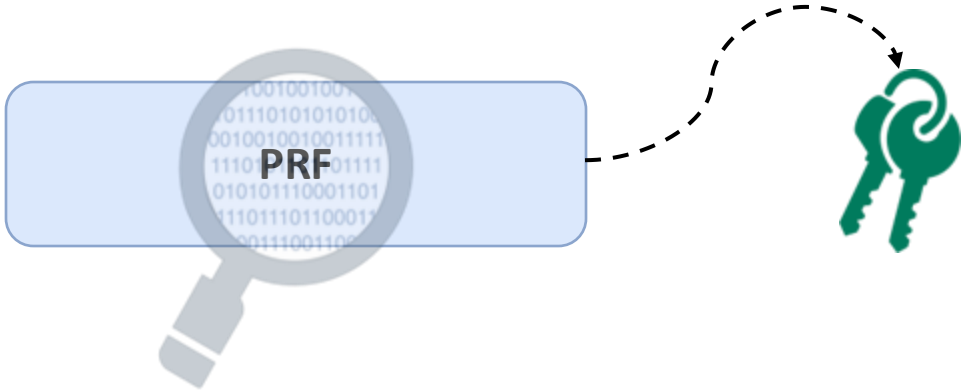
Can We Automate the Detection of Key Derivation Functions?

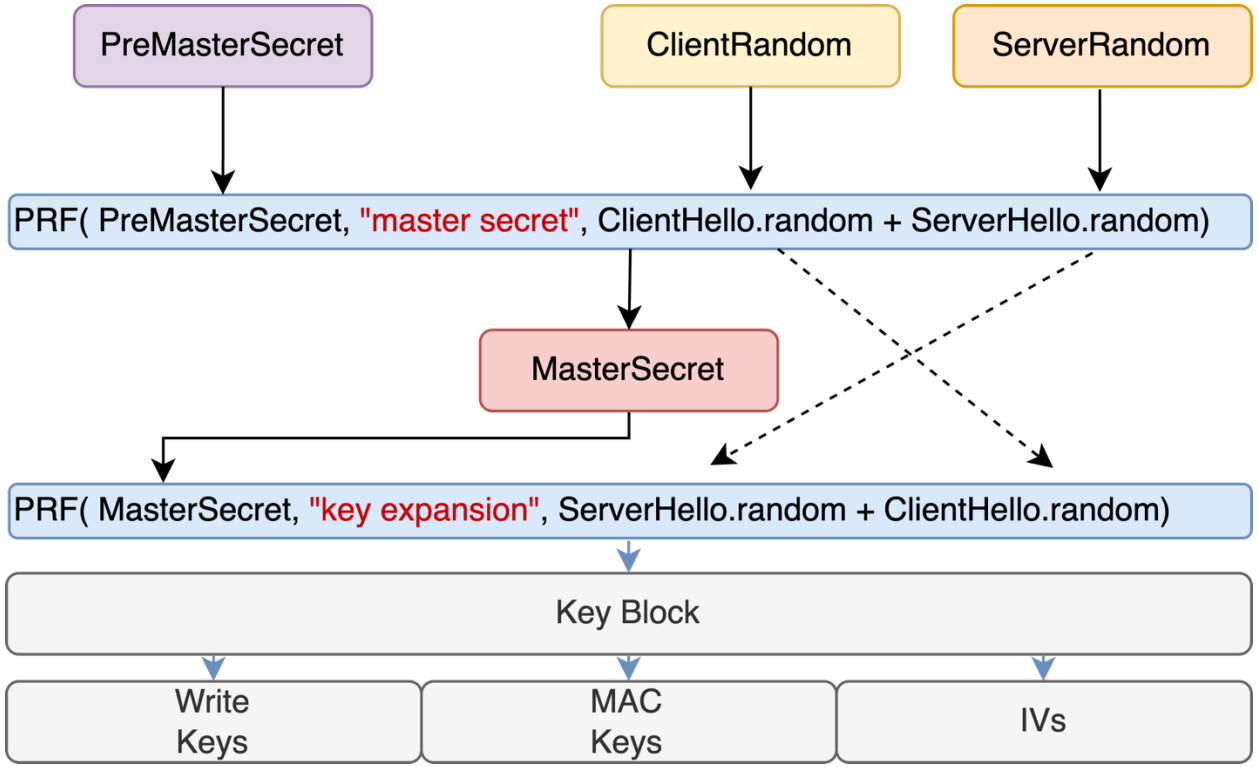
*Generated by ChatGPT

Streamlined Identifcation of TLS Key Derivation Functions

TLS 1.2

Decryption Keys TLS 1.2



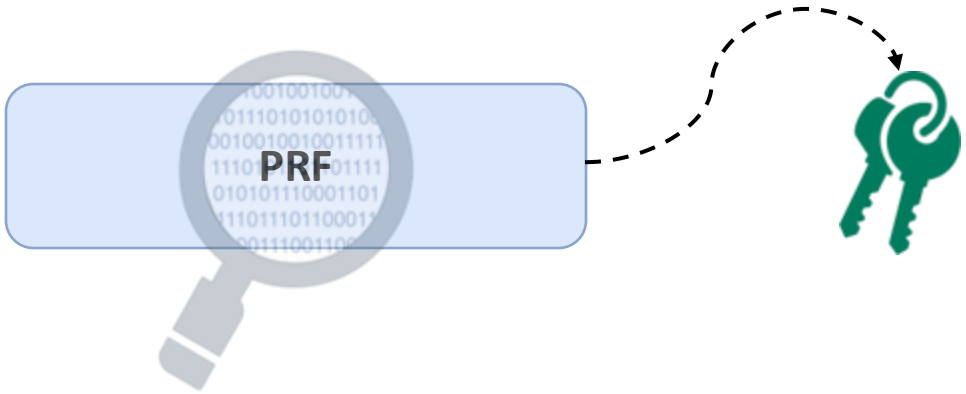


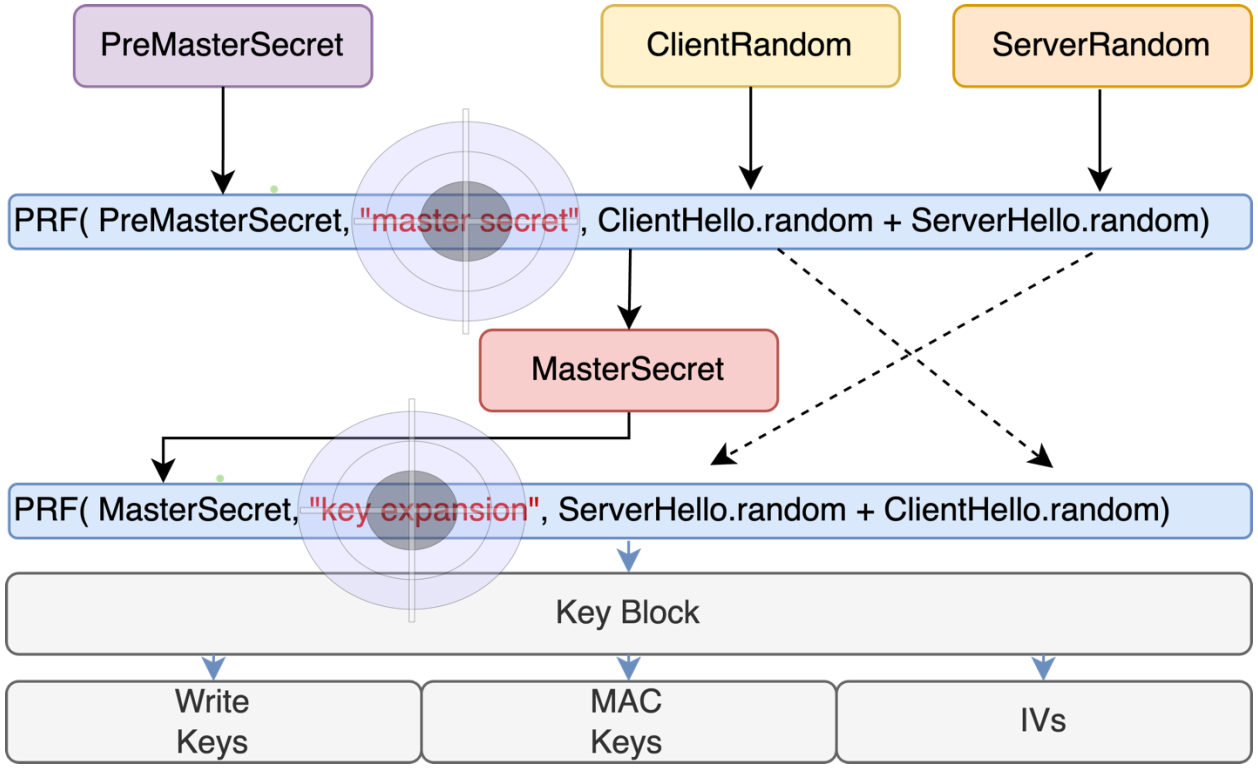
Key Derivation Function

Streamlined Identifcation of TLS Key Derivation Functions

TLS 1.2

Decryption Keys TLS 1.2



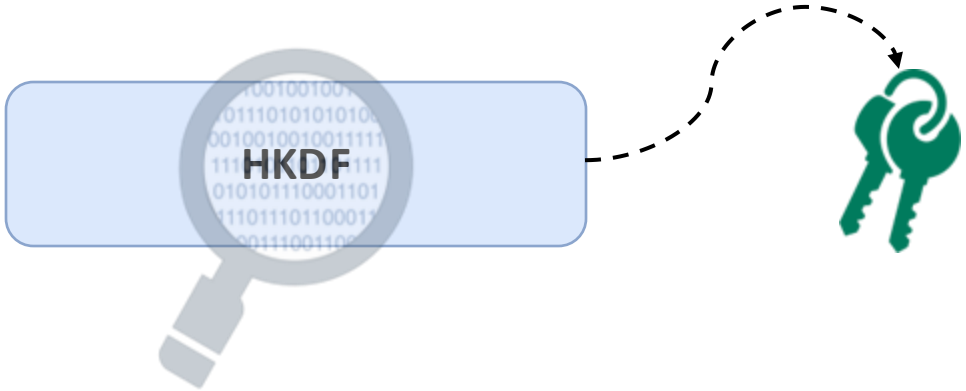


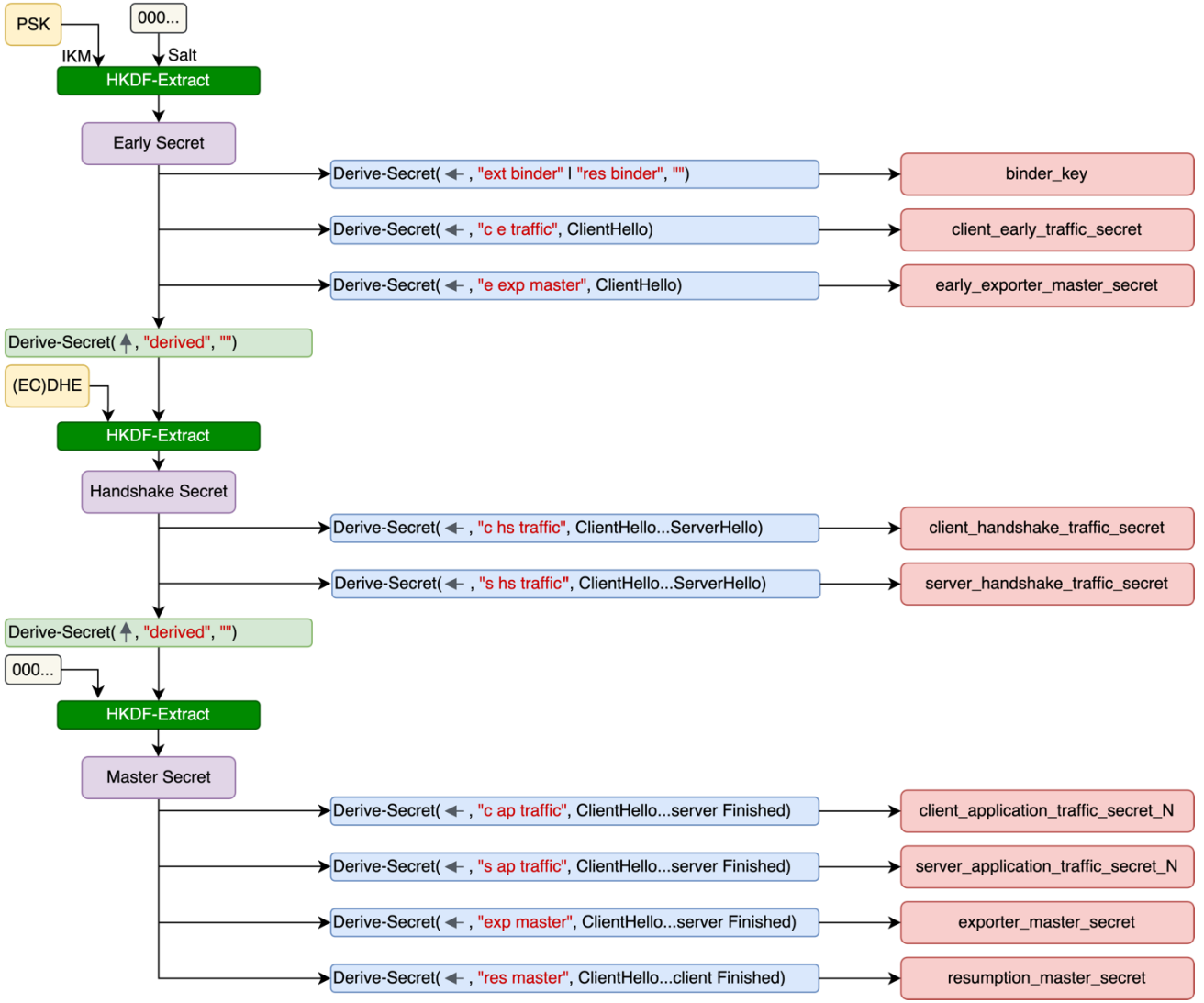
≙ Key Derivation Function

Streamlined Identifcation of TLS Key Derivation Functions

TLS 1.3

Decryption Keys TLS 1.3



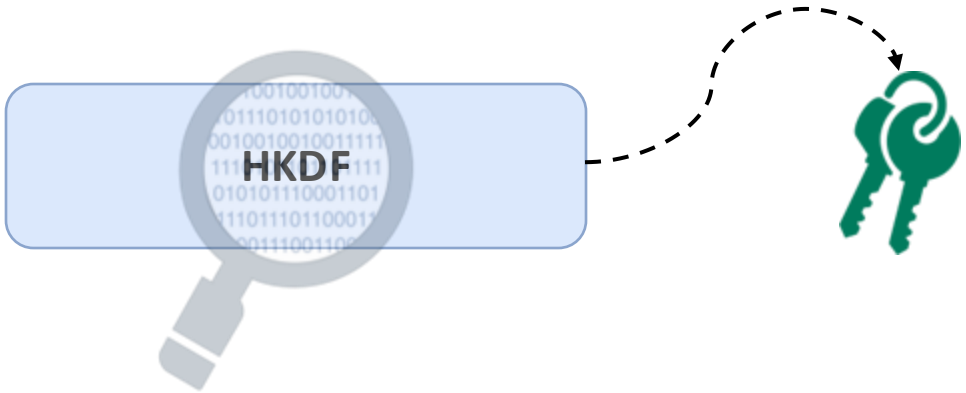


Key Derivation Function

Streamlined Identifcation of TLS Key Derivation Functions

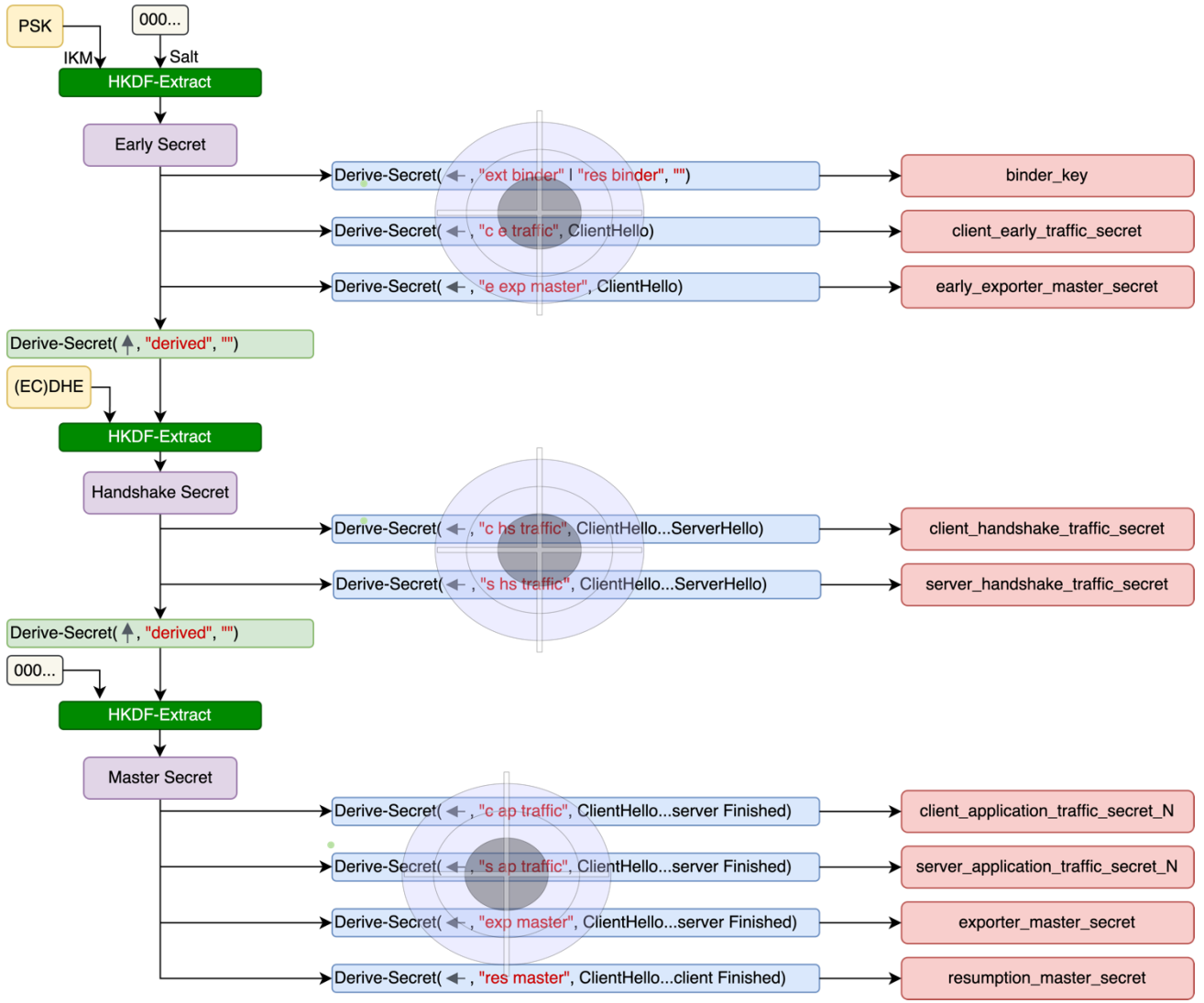
TLS 1.3

Decryption Keys TLS 1.3




```
struct {
  uint16 length = Length;
  opaque label<7..255> = "tls13 " + Label;
  opaque context<0..255> = Context;
} HkdfLabel;

Derive-Secret(Secret, Label, Messages) =
  HKDF-Expand-Label(Secret, Label,
      Transcript-Hash(Messages), Hash.length)
```

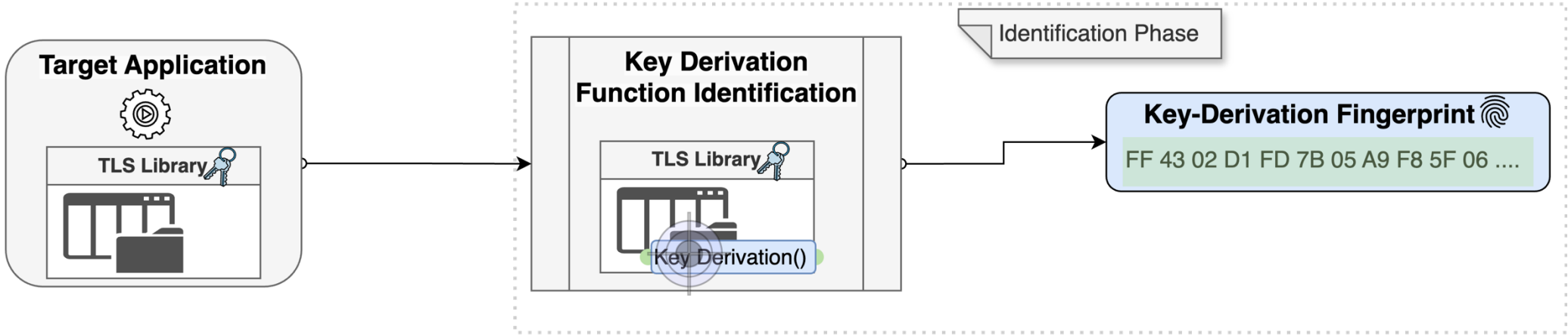


Key Derivation Function


Introducing: TLSKeyHunter
FRITap
dfrws.org
*Generated by ChatGPT

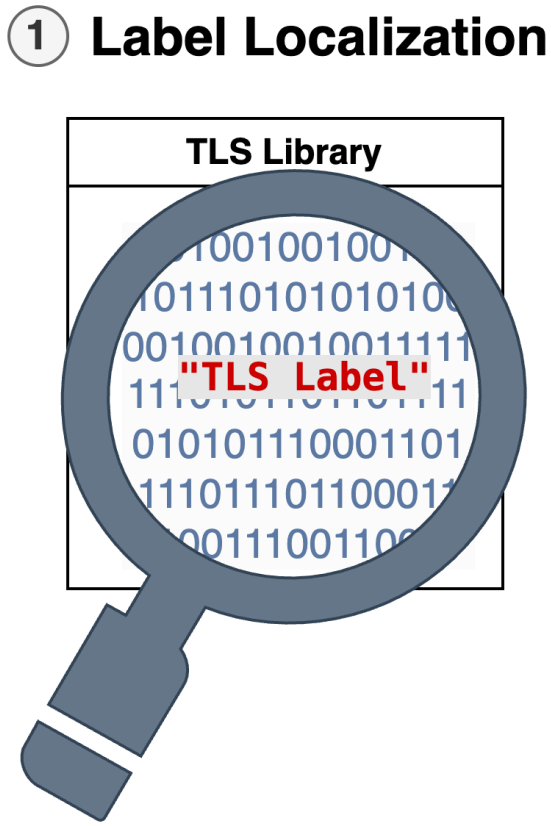
Streamlined Identifcation of TLS Key Derivation Functions

Introducing: TLSKeyHunter



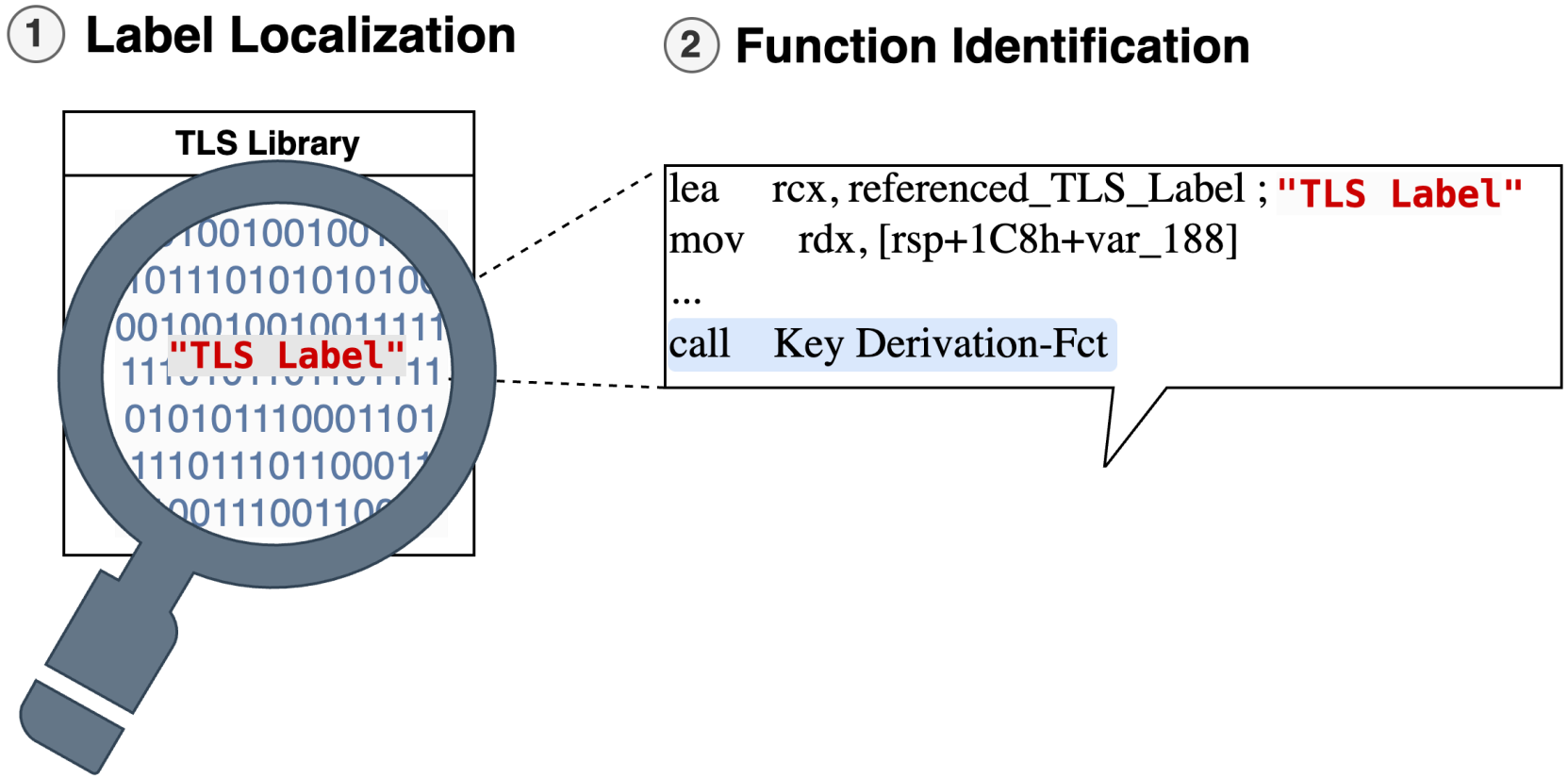
Streamlined Identifcation of TLS Key Derivation Functions

Introducing: TLSKeyHunter



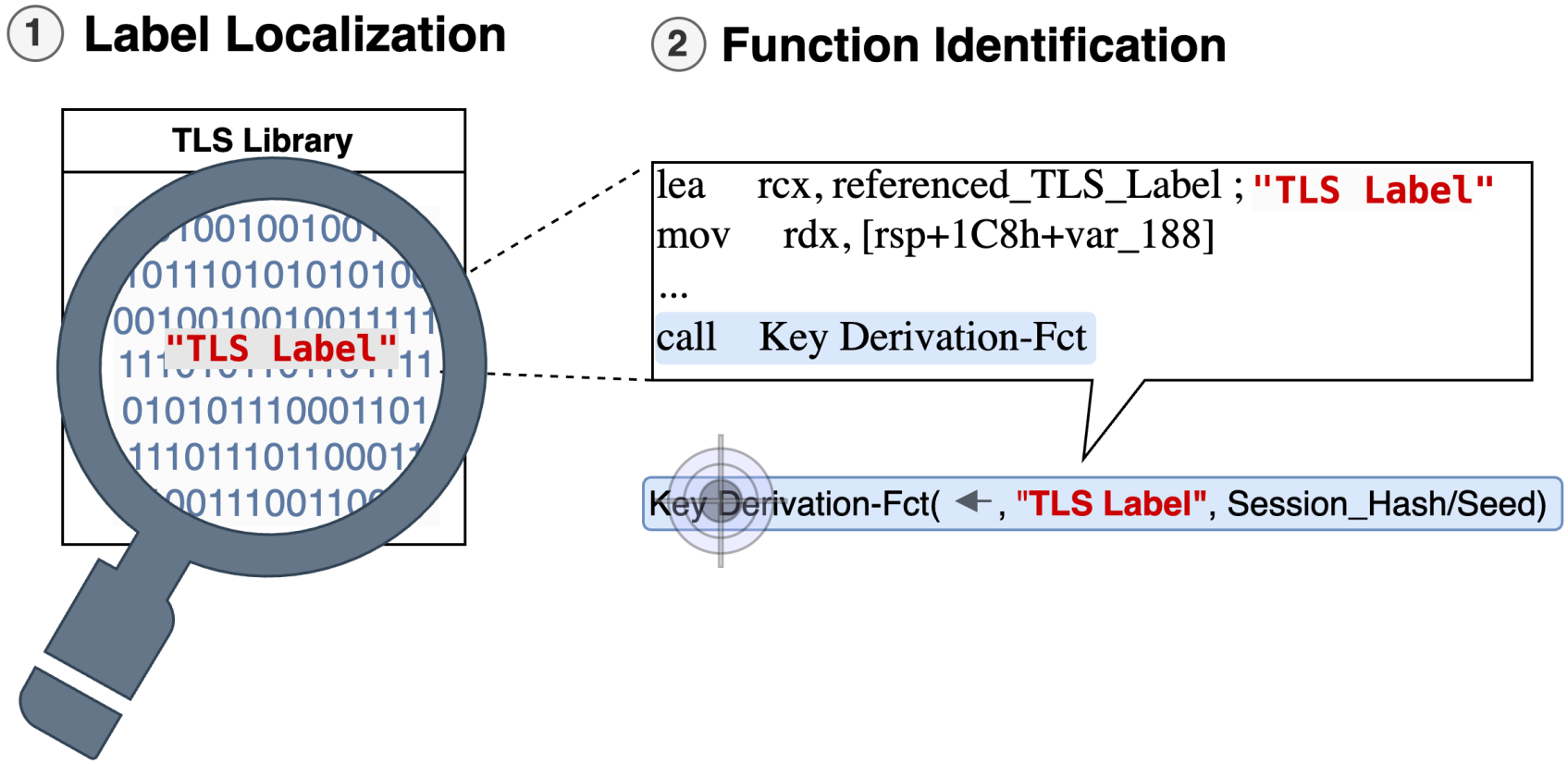
Streamlined Identifcation of TLS Key Derivation Functions

Introducing: TLSKeyHunter

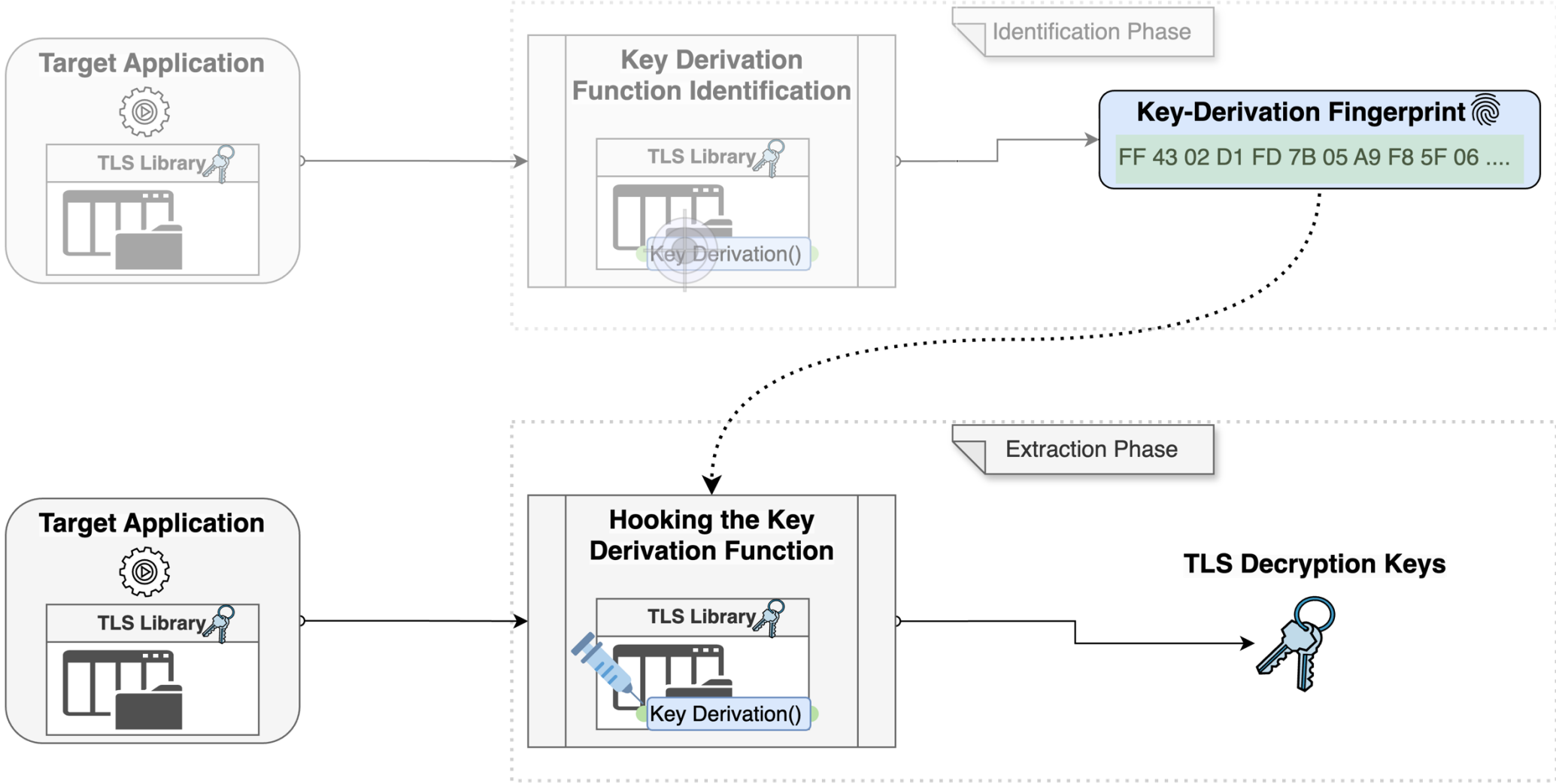


Streamlined Identifcation of TLS Key Derivation Functions

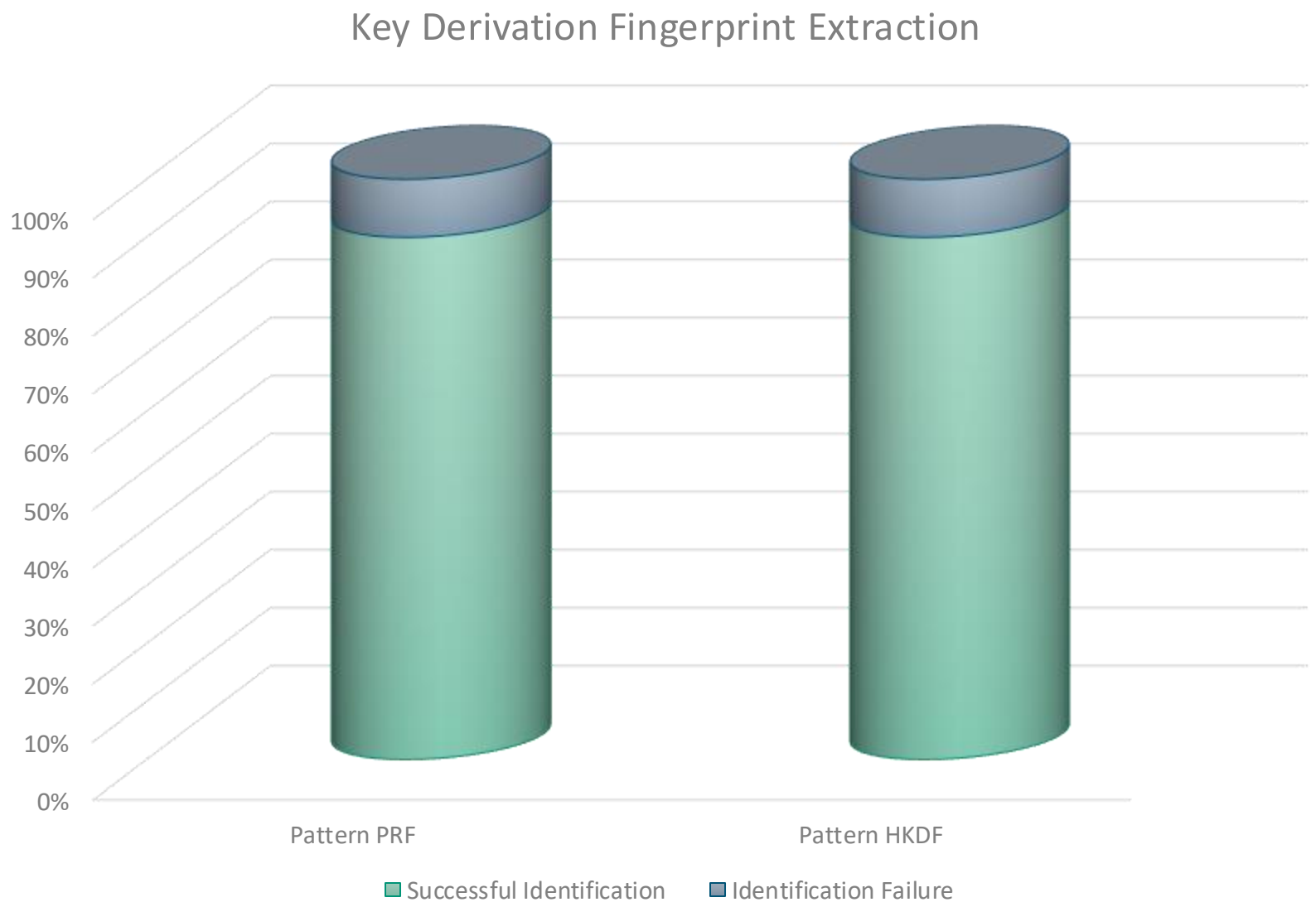
Introducing: TLSKeyHunter



Live Memory Forensic Key Extraction



Preliminary Results (Ongoing Evaluation): Key Derivation Identification



Preliminary Results (Ongoing Evaluation): Key Derivation Identification

Key Derivation Fingerprint Extraction

TLS Libraries	PRF Pattern	HKDF Pattern
Botan SSL	✓	✓
BoringSSL	✓	✓
Bouncy Castle	✓	✓
GnuTLS	✓	✓
Golang crypto/tls	✓	✓
Mbed TLS	✓	✓
RusTLS	✗	✗
NSS	✓	✓
OpenSSL	✓	✓
Schannel SSP	✓	✓
….	…	…

Preliminary Results (Ongoing Evaluation): Key Derivation Identification

Key Derivation Fingerprint Extraction

TLS Libraries	PRF Pattern	HKDF Pattern
Botan SSL	✓	✓
BoringSSL	✓	✓
Bouncy Castle	✓	✓
GnuTLS	✓	✓
Golang crypto/tls	✓	✓
Mbed TLS	✓	✓
RusTLS	✗	✗
NSS	✓	✓
OpenSSL	✓	✓
Schannel SSP	✓	✓
....	...	...

Six libraries remain to be evaluated…

Preliminary Results (Ongoing Evaluation): Key Derivation Identification

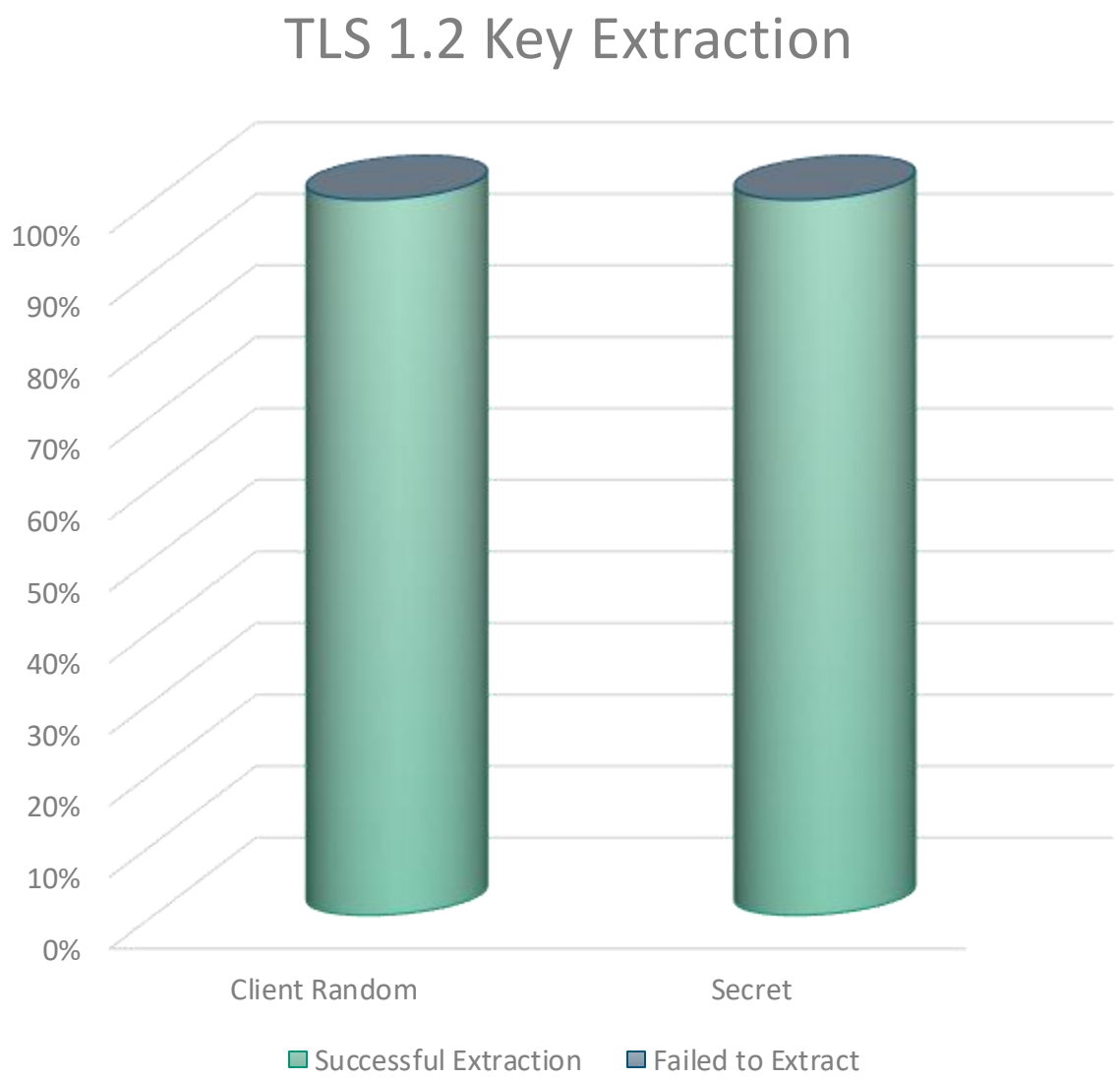
Key Derivation Fingerprint Extraction

TLS Libraries	PRF Patter	
Botan SSL	✓	
BoringSSL	✓	
Bouncy Castle	✓	
GnuTLS	✓	✓
Golang crypto/tls	✓	✓
Mbed TLS	✓	✓
RusTLS	✗	✗
NSS	✓	✓
OpenSSL	✓	✓
Schannel SSP	✓	✓
....	...	...

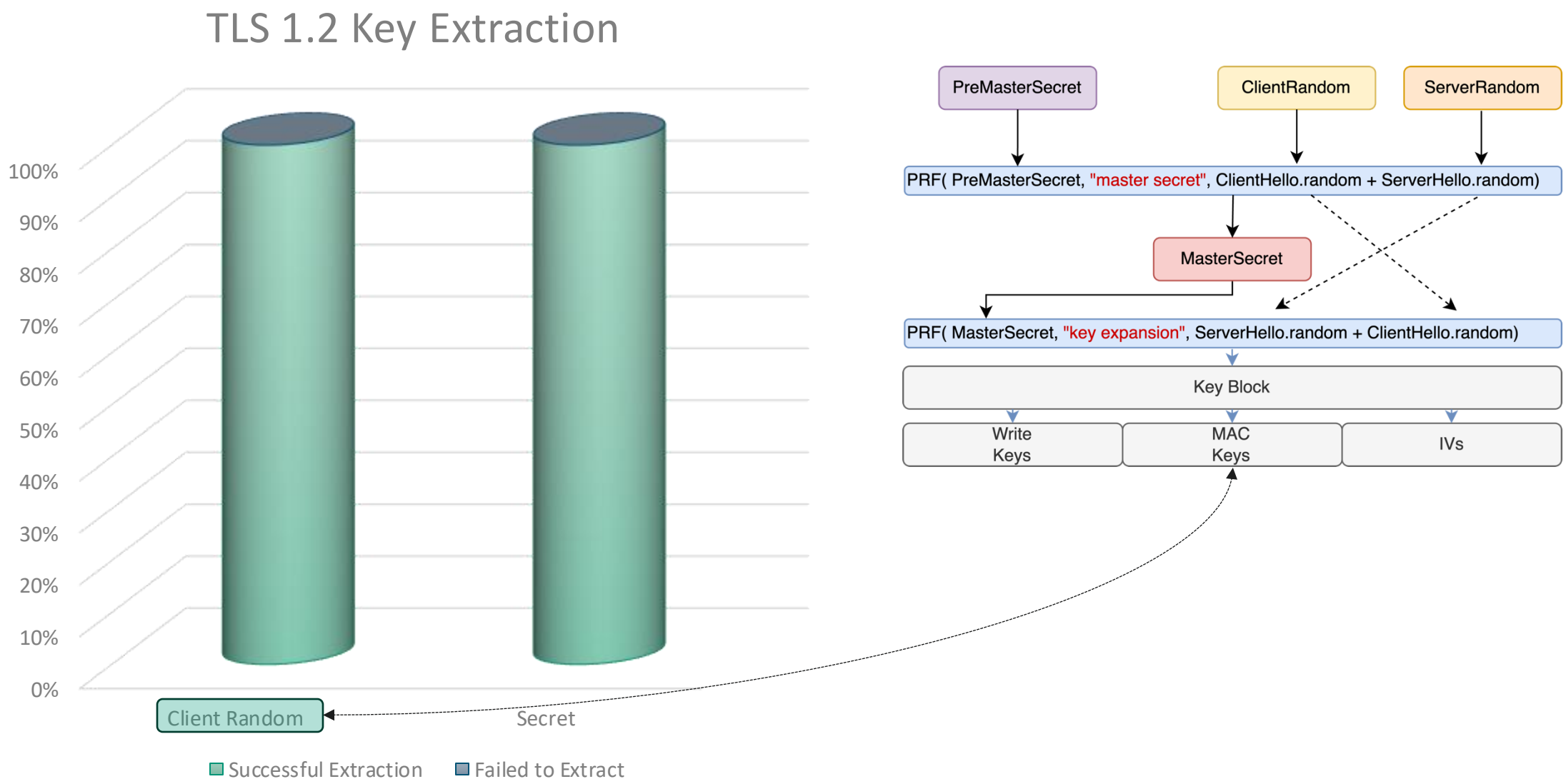
Six libraries remain to be evaluated...

Rust support intro-duced some issues, but manual verification confirmed our approach.

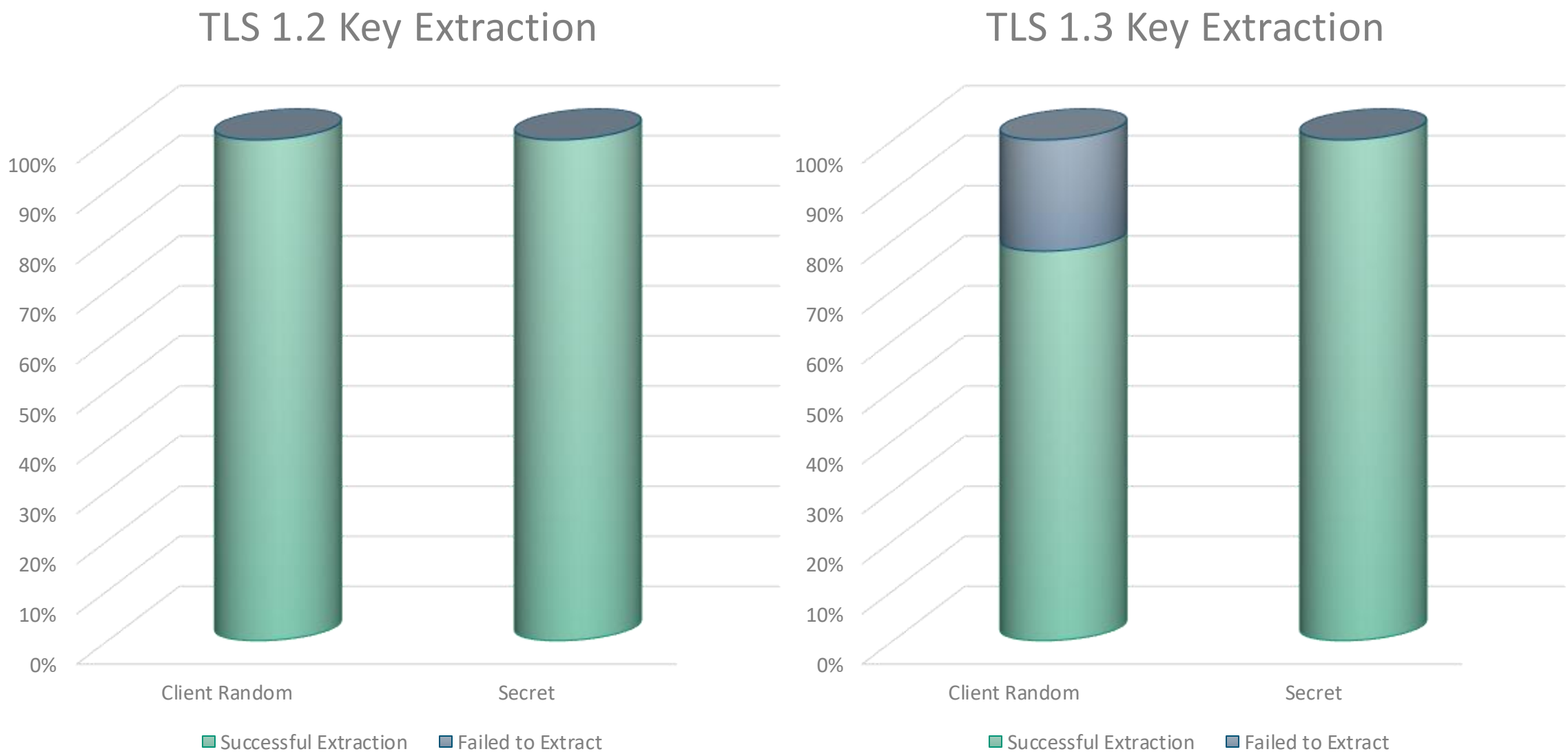
Preliminary Results (Ongoing Evaluation): TLS Key Extraction



Preliminary Results (Ongoing Evaluation): TLS Key Extraction



Preliminary Results (Ongoing Evaluation): TLS Key Extraction



*Generated by ChatGPT

Conclusion

We successfully showed **that…**

- ✓ TLS Labels offer a good anchor to identify the TLS key derivation functions
- ✓ Intercepting (hooking) them always results in the keys needed

Outlook

- Finishing our research Paper
  - Finish writing
  - Test approach against common applications
- Publishing our Framework
- Integration into FRITap

*Generated by ChatGPT

Thank you for your attention!

Daniel Baier
Cyber Analysis & Defense
daniel.baier@fkie.fraunhofer.de

Fraunhofer Institute for Communication,
Information Processing and Ergonomics FKIE
Zanderstrasse 5 | 53177 Bonn

www.fkie.fraunhofer.de