Security Processor with Quantum Key Distribution

Thomas Lorünser, Edwin Querasser, Thomas Matyus, Momtchil Peev
Austrian Research Centers GmbH – ARC
Smart Systems Division
Donau-City-Strasse 1, 1220 Vienna, Austria.
email: {Thomas.Loruenser}@arcs.ac.at

Johannes Wolkerstorfer, Michael Hutter, Alexander Szekely
Graz University of Technology, IAIK
Inffeldgasse 16a, 8010 Graz, Austria.
emails: {Johannes.Wolkerstorfer,Michael.Hutter,Alexander.Szekely}@iaik.tugraz.at

Ilse Wimberger, Christian Pfaffel-Janser, Andreas Neppach
Siemens AG Österreich, Program and System Engineering (PSE)
Gudrunstrasse 11, 1101 Vienna, Austria.
email: {Ilse.Wimberger}@siemens.com

***Abstract*—We present a fully operable security gateway prototype, integrating quantum key distribution and realised as a system-on-chip. It is implemented on a field-programmable gate array and provides a virtual private network with low latency and gigabit throughput. The seamless hard- and software integration of a quantum key distribution layer enables high key-update rates for the encryption modules. Hence, the amount of data encrypted with one session key can be significantly decreased. We realise a highly modular architecture and make extensive use of software/hardware partitioning. This work is the first approach towards application of a new key distribution technology in dedicated security processors. In particular, it elaborates requirements for the integration of quantum key distribution on a chip level.**

I. Introduction

Virtual private networks (VPN) are frequently used to connect intranets of business bodies over external and public infrastructure. A tunnel is established between the two connection endpoints and the data passes through, fully transparently to the user. To guarantee integrity and confidentiality during the transport over the public network the data have to be encrypted and authenticated.

The *Internet Protocol security* (IPsec) suite of protocols [1] is a framework providing such secure gateway-to-gateway communication. IPsec operates on network layer 3, which means that all data exchanged over the IP protocol like email (IMAP, SMTP), web (HTTP), etc. get encrypted over the tunnel transparently for the upper protocol layers, which significantly simplifies the integration into existing network infrastructures.

IPsec usually makes use of asymmetric cryptography for key agreement. In this work, we explore the feasibility to add symmetric keys agreement by means of *quantum key distribution* (QKD) in IPsec. QKD is an emerging field in security engineering. The technique was invented in 1984 by C. Bennett and G. Brassard [2] and it enables information theoretically secure distribution of cryptographic random strings. The security of the key distribution is based on the laws of quantum physics, which forbid an eavesdropper to tamper the transmission without being noticed. This introduces a new quality for key distribution and motivates for integration in security communication systems. Ideally, QKD is combined with an One-Time-Pad cipher as encryption scheme, but due to the limited key rates achieved by QKD it is necessary to combine it with an existing symmetric encryption primitive. The resulting hybrid crypto systems satisfy the bandwidth needs of modern communication systems, particularly within metropolitan area networks (MAN), and benefit from QKD as key distribution system.

We developed a highly integrated prototype which combines QKD and a hardware accelerated IPsec implementation in a system-on-chip (SoC). The prototype is realised in a field-programmable gate array (FPGA) and was demonstrated to operate together with a quantum optical setup provided by the University of Vienna. To our knowledge, this is the first time that classical cryptography technologies are combined with quantum key exchange on such an advanced level. The presented system exhibits many new and innovative features, whereby this paper focuses on the overall architecture of the SoC and further describes the efforts made to fully integrate QKD in the prototype. Moreover, it discusses the developed interfacing concepts and identifies quantum protocol dependent and generic components. The prototype will serve as a basis to analyse the requirements of QKD integration and its interaction with classical techniques.

The remainder of this article is structured as follows: Section II presents the overall architecture of the developed system-on-chip. Implementation details and results of the subsystems are given in Section III. The quantum key distribution hardware and software components as well as an interface

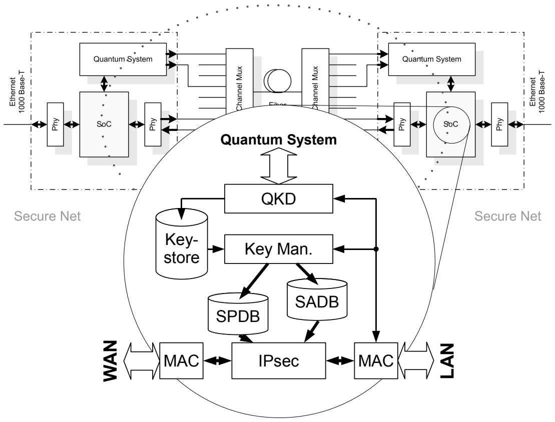

Fig. 1. Overview of the architecture and system components.

analysis is presented in Section IV.

II. Architecture

A. Overview

The system architecture of the SoC is shown in Figure 1. It is composed of four main subsystems, namely a QKD component, a key manager, an IPsec engine and a system management module.

The QKD subsystem is responsible for the generation of key material. The key stream is pushed into a keystore, which resides in memory and holds the available keys. The QKD segment can be seen as an autonomous subsystem that handles all tasks necessary to establish truly random and information-theoretically secure shared secrets between the two parties Alice and Bob.

The key-management system supervises the keystore and establishes session keys for the encrypted tunnel when required. In order to keep the keystores in sync, all operations of Alice and Bob that depend on the key material have to be done synchronously. Thus, they have to be both coordinated and atomic.

The IPsec implementation is responsible for managing the VPN connections. As specified in the RFCs, it holds a security policy database (SPDB) which defines the encryption and filtering rules for the tunnels. Active tunnels are tracked in the security association database (SADB), which buffers the corresponding keys over the defined lifetime. Lifetimes can be defined by means of duration in seconds or data in bytes. Although the IPsec standard defines various protocols, the prototype does not support all of them. In our work, we decided to concentrate on an efficient and contemporary subset. Therefore, we implemented only the encapsulating security payload (ESP) [1] protocol of IPsec, which is the more preferred standard to set up gateway-to-gateway VPNs. ESP in tunnel mode encrypts whole IP packets including the original header information and provides therefore data confidentiality. By adding authentication information, data-origin authentication of the original IP packet is achieved. From the IPsec suite of symmetric cryptographic algorithms we only implemented AES because the support of DES and Triple-DES is of retrospective interest only.

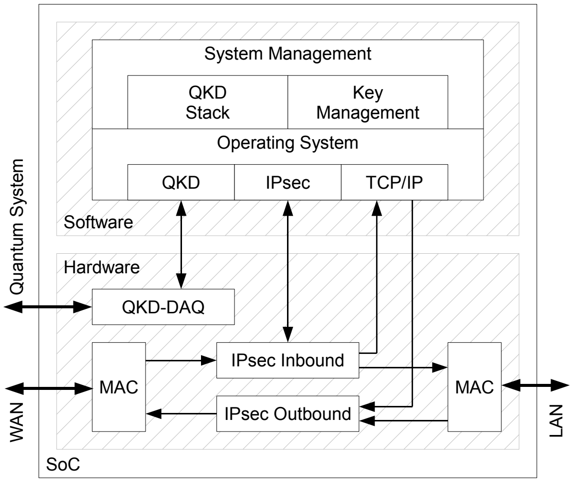

Fig. 2. System configuration.

The overall control of the system is done by a management module, which can configure, start, stop, reset and monitor the main operational modules. A WBEM/CIM [3] server was integrated, which provides a standardised way of system configuration. Moreover, it also enables remote configuration and integration in an unified management environment.

B. Hardware/software partitioning

The system is tailored in software and hardware parts as shown in Figure 2. Main parts that have been realised in hardware are the acquisition and pre-processing of the QKD as well as the filtering and cryptographic handling at the packet level for the IPsec tunneling.

For QKD, the partitioning between software and hardware is given by the communication needs and connectivity of the various stages. All modules, which require a full duplex classical communication channel, run in software and the interfacing to the external photonics as well as the pre-processing are implemented in hardware.

To achieve the required gigabit throughput and small latencies the security-gateway functionality also has to be handled by hardware. Moreover, it is not sufficient to accelerate the computational intensive task in a co-processor, instead we had to implement a complete IPsec protocol engine. It can filter, encrypt and authenticate the packets at wire speed without CPU or bus interaction. The most complete FPGA implementation of IPsec till now was done in 2005 by J. Lu and J. Lockwood [4]. They used reconfigurable resources (VirtexIIPro) to encrypt and authenticate the network traffic, but they did not filter in hardware nor did they use an operating system for seamless software/hardware integration over well established interfaces. Another argument for a hardware IPsec implementation was given by security considerations. Hardware modules are preferred in security critical applications because of better prevention from denial-of-service attacks.

III. Implementation

A. Prototyping platform

The prototype is realised on an FPGA-based prototyping platform from Xilinx. The ML410 board offers a medium-sized FPGA and has numerous interfaces [5]. The central component of the ML410 board is a Virtex-4 FX60 FPGA. It provides plenty of configurable hardware resources (25,280 slices; 232 18-kb BRAMs) and two hardcore PowerPC CPUs, which allow the creation of a real system-on-a-chip that combines both hardware and software. The CPU can be clocked up to 300 MHz and delivers a performance of up to 500 Dhrystone MIPS. Additionally, the FPGA offers four hardmacro Ethernet MACs two of which are connected to Ethernet PHYs on the board.

In order to implement the complex software architecture an operating system is needed on the CPU. We employ GNU/Linux as operating system that is provided as open source and available for the PowerPC. It already supports the important peripheral components on the ML410. The Linux kernel version 2.6 comes with a solid TCP/IP stack and builtin IPsec support. It can be modified and extended according the user needs, which is essential in our case.

B. IPsec engine

The complete packet filtering and ESP processing has been implemented in hardware. The software only configures and controls these hardware modules. The IPsec hardware supports 128-bit AES for encryption [6] and operates in the cipher-block-chaining (CBC) mode. For authentication, AES-XCBC-MAC-96 is used, which is part of the cryptographic suites for IPsec [7].

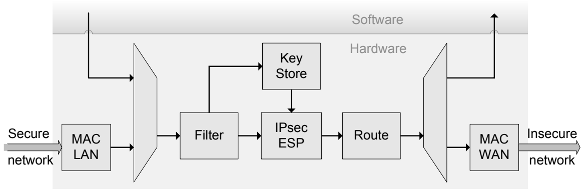

Fig. 3. IPsec processing module.

The hardware integrates two independent modules for both inbound and outbound IP traffic. In Figure 3, the processing of outbound traffic is shown. The inbound module is not discussed in detail but operates in a very similar manner. The data stream of the Gigabit Ethernet is received by a media-access controller (MAC) which is realised as a hardmacro on the FPGA. The MAC converts the data stream into a byte stream and some control signals. This 8-bit bus is clocked at 125 MHz and is used throughout the IPsec modules. All components of the outbound path operate in the same clock domain of 125 MHz. The modules do not process whole Ethernet frames, which can contain up to about 9,000 bytes. Instead, they work in a pipelined fashion and start processing a frame immediately after receiving the relevant header data. This guarantees short latencies and avoids large frame buffers.

Frames are received either from the MAC on the secure network interface (LAN) or from the software. The filter module then decides what action has to be applied to the frame. The action can be: bypass, discard or protect. In the case of protect, the ESP module encrypts and authenticates the frame with keys provided by the QKD system. If the destination of a frame is the gateway itself, it is transferred to the software. Otherwise, the routing module retrieves the correct destination MAC address and forwards the frame to the MAC on the insecure WAN interface.

The IPsec engine, including the SADB and SPDB tables, needs 9,215 slices and introduces an extremely low packet-latency of only 2,312 ns.

C. Key management

The management of the QKD generated session keys is implemented in software. The software consists of two components: a key manager and a modified version of the Internet Key Exchange protocol (IKE). The key manager distributes the stream of symmetric quantum-keying material originating from the underlying QKD subsystem into session based key buffers. The negotiation for the session key buffer parameters is based on the IKE. IKE is used to establish a symmetric session key that can be used in IPsec. Typically, it is based on the Diffie-Hellmann key exchange protocol. In our implementation, we have replaced all asymmetric algorithms by using keys generated by the QKD device. Furthermore, additional parameters are introduced to negotiate, for example, QKD key rates and buffer sizes for the session-based QKD keystores. Using the negotiated parameters, the key manager establishes the requested QKD keystores and the IKE starts to generate so-called security associations (SAs) that are passed to the IPsec engine using the standard PF_KEY interface. SAs typically contain parameters like encryption, authentication algorithms, session keys and lifetime of the keys in seconds and/or bytes. If a key expires, a notification is sent to the IKE daemon that provides a new SA and synchronizes the IKE daemons on both gateways. Thus, lossy connections can be compensated and session key buffers can be resynchronized.

D. System management

In order to provide an appropriate end-to-end management solution for our system, we decided to follow the paradigm of the Web-based Enterprise Management (WBEM) [3] initiative. The main objectives of this initiative have been to provide a set of standards to allow interoperability among multiple-vendor management products. Therefore, the Distributed Management Task Force (DMTF) has defined several standard models such as the Common Information Model (CIM). CIM allows the description of a networked device in a totally object-oriented manner. However, as a management server application we have decided to use the Small Footprint CIM Broker (SFCB) [8] that has been especially developed for resource constrained devices. We have implemented various CIM providers that allow the configuration and monitoring of many system parameters of the prototype as needed by QKD keystore, QIKE, IPsec and

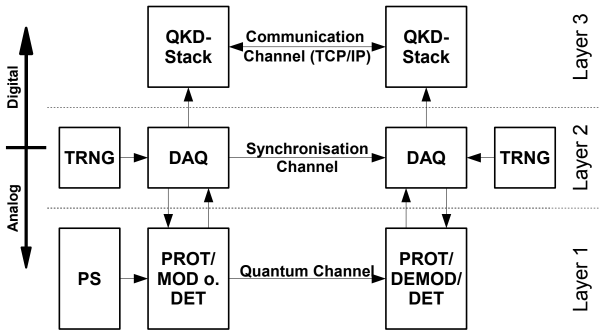

Fig. 4. Overview of a QKD system.

networking module. As a client application, we have implemented a graphical user interface (GUI) that is able to establish a connection to the SFCB server over HTTP using Secure Socket Layer (SSL) as well as provide basic authentication. The transferred data itself is encoded using CIM-XML which is currently the only standard based protocol for exchanging CIM information. The client-server architecture allows an easy and convenient system administration that has not only been customized for our individual use but offers a global end-to-end manageability that can be accessed by every standardized WBEM/CIM-compliant administration tool.

IV. QKD SUBSYSTEM

A. *Function and interfaces*

In general, a quantum key-exchange system provides provably secure distribution of random secrets over a quantum channel. A quantum channel is an optical transmission channel like a fiber link or a free-space line of sight connection. A complete QKD system consists of a quantum optical segment, a data acquisition system (DAQ) and a processing segment. The latter requires an authenticated classical communication channel for final key distillation which does not need to be confidential. When applied together with One-Time-Pad, QKD is often referenced to as quantum cryptography. The quantum part is built out of photonic components like photon sources (PS), transmission systems, modulators (MOD) and single-photon detectors (DET). There exist various encoding and transmission schemes (PROT) to realize a QKD system, but nearly all of them use the same basic building blocks. Additionally, all implementations heavily rely on the capability of time-synchronous event measurement.

An overview of the basic building blocks for QKD systems is shown in Figure 4. The system can roughly be divided into 3 layers. The photonics can be referred to as a quantum optical layer. It contains all components to send qbits from Alice to Bob. The electrical signaling to interface with layer 1 has still to be considered as analog, although it uses standard digital signal levels. This is due to the fact that the information lies in the precise timing of events for modulators and detectors.

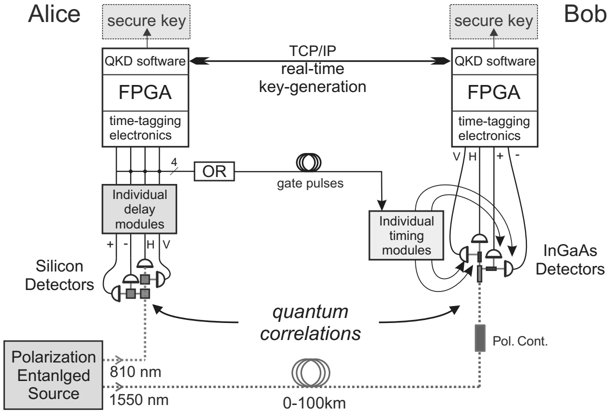

Fig. 5. Quantum optical layer.

Layer 2 is responsible for the correlated generation and detection of qbits, which leads to the so called raw key. The raw key represents correlated binary data generated on both sides of an QKD link which can be further processed to the shared secret. To achieve this task the two key generation parties need to be synchronised. This is typically done over an additional synchronisation channel. Due to the loss of the photons over the quantum channel, the clock cannot be recovered out of the signal itself, as done in classical communication systems. Additionally, the layer 2 often needs input from an ideal entropy source (TRNG - true random number generator) if not provided by the layer 1.

The further processing, which requires a classical communication channel can be subsumed in a third layer referred to as QKD stack. Except the initial sifting stage the remaining protocol stages of the QKD stack can be considered as system independent. Sifting still depends on the QKD protocol applied because it ”knows” how to generate the raw key which in turn depends on the particular coding scheme applied.

B. *Time correlated data acquisition*

Two layer 2 DAQ modules were developed and tested for use with fiber coupled QKD systems based on polarization entanglement. These types of systems are being designed in various university labs and the University of Vienna delivered a fully operational quantum optical layer for testing purposes. However, with minor modifications the DAQ modules can also be applied to other QKD systems.

The entangled system applies a modified BB84 protocol [9] and is fully compatible with the developed system-on-chip. An overview of the resulting QKD system is depicted in Figure 5. It should be mentioned that in contrast to the majority of QKD systems it has a built in TRNG and works passively without need for external modulation.

A source of entangled photons and one detection unit are located at the transmission side (Alice), hence, one photon of the pair is immediately measured. The second photon is transmitted to the distant location (Bob), where it is analysed in the second detection module. To get the best efficiency it is important to measure the detection times with high accuracy. Ideally the resolution of the measurement system is in the order of the jitter of the detectors. Depending on the applied type it is in the range of 50-300 ps.

One version of the measurement core fully embeds layer 2 in the FPGA. We used the high-speed serializer/deserializer integrated in the Xilinx Virtex-4FX devices as hard core (MGT - Multi Gigabit Tranceiver) to sample the photon counting events. In this way we achieved a resolution of 333 ps with the MGTs running at 3 Gbps. The measurement core is capable of handling highest repetition rates. It was tested up to 70 MEvents/s but is by far not limited to it.

The second DAQ core that has been developed, interfaces to an external time-to-digital converter (TDC) and handles up to 40 MEvents/s at a resolution of 80 ps. This version of the DAQ unit was selected for integration into the presented prototype due to the better timing resolution and higher flexibility in the lab. However, the MGT version seems to be ideal for real products because it reduces external circuitry to a minimum.

After time stamping, the DAQ core also pre-processes the events. It converts the stamps into system time and filters the stream for interesting events in a trigger-matching unit. Finally, the raw key stream is passed to the software driver, which stores the data in a large buffer. The DAQ core uses about 3000 slices, which is about 11% of the overall resources present in the Virtex-4 FX60 device. The core uses one clock domain running at 100 MHz and connects to the PowerPC via the SoC bus.

C. Key distillation

In layer 3 (QKD stack) the data are sifted, error corrected, privacy amplified and pushed into the keystore. These stages are implemented in software and run on the embedded PowerPC. The Linux TCP/IP stack is used for public communication. The messages are sent in plain text but authenticated. The software stack is programmed under heavy usage of Linux/Unix multi-processing and interprocess capabilities to optimize CPU performance. During the sifting phase useless entries in the raw key are eliminated. The data are then passed to the error correction stage, which runs a performance optimized version of the Cascade protocol [10]. Optionally, the stack supports low density parity check (LDPC) codes as an error correction scheme. After correcting all errors and confirmation of the guaranteed bit error rate, the stream enters the privacy amplification stage. This module shrinks the keys according to the amount of information revealed in the public conversation and the detected error rate. The privacy amplification is in principle a parametric hash function based on the universal$_2$ class of hash functions [11][12]. To prevent man-in-the-middle attacks all communication has to be authenticated and checked for integrity. This is done by applying information-theoretically secure message authentication and, in particular, by implementing the evaluation hash [13].

The overall key rate that is achievable with the software implementation of the QKD stack is limited by the computing power of the PowerPC which delivers a rather poor computing performance. After optimisation of algorithms and protocols we show rates for final keys up to 3 kbps and raw key performance up to 10 MEvents. Even though this is not the fastest system compared to PC implementations, it is well suited for the overall SoC. It handles the typical raw key-rate of the optical layer for distances from 20 to 50 km. This provides enough key material for frequent key-refreshment in the IPsec engine.

D. Implementation considerations

Our implementation shows a high level of integration and as such necessarily incorporates quantum protocol dependent components. This reduces the need for external circuitry but also reduces the possibilities for universal usage and distribution. This is not relevant for a realisation in programmable logic, but holds for ASIC developments. For the development of QKD specific ASICs it is desired to have a standardised raw-key interface and a dedicated but external physical DAQ system. This is analogous to the separation of the physical layer and the link layer in the Ethernet standard. Moreover, Ethernet introduces a set of standardised media independent interfaces, which allow seamless connection of different media to the link layer. Such a concept seems also to be reasonable for quantum key distribution and may result in broader application of this technology.

V. Conclusion

In this work, we present a prototype for a QKD-enabled security processor. It is implemented on an FPGA rapid prototyping platform and is fully functional. Together with an entanglement-based quantum optical layer, it builds a complete system, being the first one integrating quantum key distribution with high speed encryption and frequent key-update in one chip. Moreover, the IPsec packet engine is the first full protocol implementation in an FPGA. The overall system fits in a medium sized FPGA and has potential for upscaling to higher throughput and higher key generation rates. The highly modular approach makes this prototype an ideal reference system for further investigations of deployment and performance scenarios around hybrid QKD-enabled security processors.

Additionally, we developed an unified QKD system architecture in a structured approach. Further specification and standardisation of generic interfaces will be the basis for a wider adoption in security and networking industries. The tight integration of all digital components in a single chip simplifies the applicability of QKD. It makes QKD a new building block for system manufacturers and facilitates its distribution.

Acknowledgement

This work origins from the Austrian Government funded project *QCC* (Project 810195) established under the system-on-chip programme of FIT-IT.

References

[1] S. Kent, “RFC 4303: IP Encapsulating Security Payload (ESP),” RFC 4303 (Proposed Standard), Dec 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4303.txt

[2] C. H. Bennett and G. Brassard, “Quantum Cryptography: Public Key Distribution and Coin Tossing,” in *Proceedings of International Conference on Computers, Systems and Signal Processing*, Dec 1984.

[3] DMTF, "Web-Based Enterprise Management (WBEM)," http://www.dmtf.org/standards/wbem. [Online]. Available: http://www.dmtf.org/standards/wbem

[4] J. Lu and J. Lockwood, "IPSec Implementation on Xilinx Virtex-II Pro FPGA and Its Application," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium – IPDPS'05*. IEEE, 2005, p. 158.2.

[5] Xilinx, "Virtex-4 FX ML410 Embedded Development Platform," Feb 2007. [Online]. Available: http://www.xilinx.com/ml410

[6] National Institute of Standards and Technology (NIST), "FIPS-197: Advanced Encryption Standard," November 2001. [Online]. Available: http://www.itl.nist.gov/fipspubs/

[7] P. Hoffman, "RFC 4308: Cryptographic Suites for IPsec," RFC 4308 (Proposed Standard), Dec 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4308.txt

[8] Standards Based Linux Instrumentation, "Small Footprint CIM Broker (SFCB)," http://sblim.wiki.sourceforge.net/Sfcb. [Online]. Available: http://sblim.wiki.sourceforge.net/Sfcb

[9] G. B. C. H. Bennett and N. Mermin, "Quantum cryptography without bell's theorem," *Phys. Rev. Lett.*, vol. 68, pp. 557–559, 1992.

[10] G. Brassard and L. Salvail, "Secret key reconciliation by public discussion," *Lecture Notes in Computer Science*, vol. 765, pp. 410–423, 1994.

[11] G. B. C. H. Bennett and J. M. Robert., "Privacy amplification by public discussion," *SIAM J. Comput.*, vol. 17, no. 2, pp. 210–229, 1988.

[12] Charles H. Bennett, Gilles Brassard, Claude Crépeau and Ueli M. Maurer, "Generalized privacy amplification," *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1915–1923, 1995.

[13] V. Shoup, "On fast and provably secure message authentication based on universal hashing," *Lecture Notes in Computer Science*, vol. 1109, pp. 313–328, 1996.