A Survey of Implementing Provable Security in Cryptosystems

Kyle Barton
Computer Information Technology
Purdue School of Engineering and Technology
Indianapolis, IN USA
kylbarto@iu.edu

Alston Listenberger
Computer and Information Science
Purdue University School of Science
Indianapolis, IN USA
alistenb@iu.edu

Michael Waddell
Computer and Information Science
Purdue University School of Science
Indianapolis, IN USA
mpwaddel@iu.edu

***Abstract*—Often when attacking a system, the underlying cryptographic protocols are typically thought to be the strong point in the system. This is simply because when designing systems designed to resist an attack on a cryptographic protocol, engineers follow a standard cost-benefit analysis and typically seek to make the cost of breaking into a system outweigh the rewards, thereby reducing the potential for attack. So, breaking a complex cryptographic protocol ordinarily is too prohibitive from a perspective considering time and finances. However, for numerous reasons this may not always be the case. Incorrect implementations may result in a protocol failure, but if a protocol is correctly implemented, the strength of such a protocol is typically demonstrated in the form of a mathematical proof. Provable Security is a methodology which seeks to guarantee a scheme is secure relative to the current understanding of the underlying mathematical principles inherent to the implementation of such a protocol through the usage of such a mathematical proof. With this definition in mind, one can imagine situations in which, as our mathematical understanding of a principal grows, so too does the possibility of cryptographic failures in the protocol. A comprehensive study of what it means for a system to be provably secure is presented, along with examples of protocols, which were once proven secure, eventually being demonstrably insecure.**

***Index Terms*—Provable Security, Cryptography, Cryptosystems**

I. Introduction

Cryptography is an important, yet often overlooked, facet of modern life. Nearly sharing an equal importance - and often working hand-in-hand - with physical security, cryptographic protocols provide protection to structures of critical importance like the national power grid and financial networks. Public key cryptography is the defining feature of cryptocurrencies, which are increasingly utilized for transactions as acceptance becomes more widespread. Furthermore, communications and data transfers related to national security are handled using cryptography to avoid classified information from being acquired by adversaries. The above-mentioned scenarios fail to capture the enormity that is the field of cryptography, with many differing sectors relying on a handful of encryption schemes, outlined in their standard practices, to protect their most sensitive data, a failure of one of these protocols could lead to large-scale disaster.

The failure of an encryption scheme for a medical device could place untold numbers of patients at risk at hospitals around the world. A similar failure in encryption could spill vital state or military secrets jeopardizing our national security. In banking, a broken protocol could lead to economic manipulation on the scale of trillions of dollars being incorrectly allocated. The chaos any of these events could cause are massive. With cryptography playing such a key part in the world's most vital sectors, one must be sure it will function as intended before a cryptographic protocol is applied to such a system for protection.

In order to determine the security of a protocol, we must first set certain standards for the sake of communication and comparison. A protocol can be said to have n-bit security which means that a key is 'n' bits long. This means that – on average – an attacker would manually have to compute $2^{(n/2)}$ operations to guess what the potential key is. As a key grows larger, it becomes more and more prohibitive, when considering time or resources, for an attacker to brute force a specific key. This degree of security would be referred to as computationally secure. Another degree of security is unconditionally secure, meaning that – even with infinite computational time – it cannot be broken.

Table 1 is an example of security level for varying algorithms detailed in US NIST SP-800-57 Recommendation for Key Management [1].

TABLE I
Comparing Security Strength of Different Algorithms

Security Bits	Symmetric Key	RSA	Elliptic Curve
< 81	*2TDEA	k=1024	f= 160 - 223
112	*3TDEA	k=2048	f= 224 - 255
128	AES	k=3072	f= 256 - 383
192	AES	k=7680	f= 384 - 511
256	AES	k=15360	f= 512+

Red Text signifies algorithms which are now NIST deprecated.

We can ensure that a protocol guarantees a certain degree of security through the usage of mathematical proofs. The process of proving a system is secure from attacks or failure is called provable security. Provable security works in much the same way one would prove a mathematical principle. One could attempt to prove a system is insecure by presenting an attack and demonstrating through proof it will be successful. We can also take a proof by contradiction route and make some

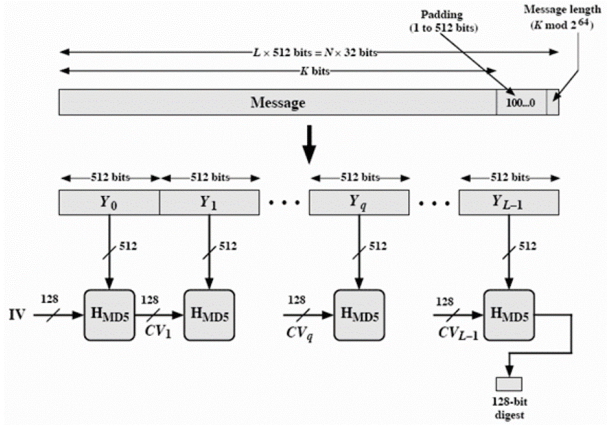

Fig. 1. The weakness of MD5 is that a collision can be abused to forge a valid digital certificate. This collision occurs because an infinite number of inputs can create only a finite number of outputs.

assumptions about the system, then use these assumptions to show that a successful attack would lead to a contradiction, thereby making the attack described impossible under the given conditions. Finally, provable security could also give a researcher a hint towards potential errors in their system if they attempt to prove it is secure and fail to do so.

The provable security of an algorithm depends heavily on thoroughly testing and reviewing an algorithm as well as the capability of computation at the time we are attempting to prove it is secure. For instance, when analysis showed the hashing algorithm MD4 was likely insecure, MD5 was designed to be a provably secure replacement to it. However, MD5 was later shown to be easily broken utilizing a simple home computer [3]. MD5 can be seen in Figure 1. Other encryption algorithms such as TripleDES, which evolved from DES, were once recommended as the standard encryption, but later were shown to be insecure and subsequently deprecated by the National Institute of Standards and Technology in 2017 [4].

If an algorithm is shown to be provably secure, there is still a possibility it can be broken. Implementation failures, in which the algorithm is improperly set up or configured, can lead to an adversarial party easily breaking the encryption. The potential for side channel attacks also exist which can bypass the proof entirely and exploit a previously unconsidered mechanism to break the encryption. This side channel attack can occur even without flaws in the underlying encryption scheme with the real weakness again being the overall implementation and not the encryption scheme itself.

Many researchers are currently engaged in the field of provable security. In this paper we examine a small section of this large field and address what it means for a system to be provably secure, and whether our current approaches to provable security are the most optimal and effective way to demonstrate security. Furthermore, we will examine provable security further by examining some protocols which are currently thought to be provably secure and have other protocols which also rely on this provable security. We also present and examine some protocols which were once shown to be provably secure only to be broken later such as the Elliptic Curve Digital Signature Algorithm - a widely used signature scheme today – broken by a side channel attack called LadderLeak [5], and Onion Routing, a technique for anonymous communication over a network - which has been exploited to remove anonymity [6]. Finally, we will conclude with our outlook on the future of provable security in cryptography.

II. Cryptographic Proofs and their Implementation

As proofs work in the field of mathematics, cryptographic proofs can apply one proof to the proof of another protocol. This creates the building blocks for future proofs and the field of cryptography itself. As the field of cyber security, and by extension cryptography, continues to grow, more proofs will begin to stack onto this foundation of proofs. In cryptography, proofs which show a reasonable degree of security are known as computationally secure proofs. Cryptography experts use the term computationally secure to mean a protocol would take too much time or computational effort to be worth attacking. But what happens in a post-quantum computing generation of cryptography? Computational effort can be accomplished more efficiently, eventually for less money, and taking much less time. When this occurs, what happens to the building blocks of cryptography? Many modern proofs will account for this by acknowledging they are only secure in a pre-quantum computing environment. Below the strengths, weaknesses, and application throughout history of Provable Security are explored.

A. Historical Use of Proofs in Cryptography

Provable Security has been used for over four decades with only a handful of noteworthy proofs which had a major flaw found only after the proof of its security was announced. Although these are major issues, it only accounts for a small percentage of all cryptosystems which utilize provable security. According to A Brief History of Provably-Secure Public-Key Encryption by Alexander W. Dent, “The first attempt to prove the security of a public-key encryption scheme was by Rabin in 1979, who described an encryption scheme for which recovering the message was as intractable as factoring an RSA modulus. Later, Goldwasser and Micali described a scheme which they could prove hid all information about the plaintext. However, it wasn’t until the early 1990s that researchers began to establish reliable and easy to use formal models for the security of an encryption scheme and that the cryptographic community began to think about constructing practical and efficient provably-secure public-key encryption schemes” [7]. From the very beginning of this methodology, Rabin had compared the security of one device to that of another which was accepted. Over the decades, this has caught attention and gained formality; however, with the rise in marketing this terminology is becoming a buzzword whose meaning is being misused.

Additionally, the current culture of professional research encourages publishing often. On top of this, researchers who want to create new methods often do not make money directly off of their research. For example, when a variant of the Rijndael block cipher was created by Rijmen and Daemen was entered into a competition created by the National Security Agency (NSA) in the late 1990's and early 2000's, it was chosen as the Advanced Encryption Standard (AES) to replace DES and DES3. For each entry, the researchers had to agree if their design won, they could not patent or profit from the design, as the goal of the chosen AES was to be widely implemented across devices and a barrier to entry was seen to be less ideal [8]. This non-exclusive, royalty-free model does not provide the most encouragement or competition when designing ciphers.

Security Proofs rely on original security assumptions for each type of cipher. ElGamal can be compared to the hardness of other Discrete Logarithms and Finite Fields. RSA and some other integer factorizations rely on the Strong RSA Assumption. Many public-key cryptosystems use Learning with Errors (LWE) as their hardness assumption. The Elliptic curve relies on its own discrete logarithm problem as well. AES, being the only system certified as secure by the NSA, is a symmetric key system; symmetric key systems are inherently un-provably secure. AES is only secure because it has survived for two decades under constant scrutiny by the smartest cryptographic minds with very few successful avenues of attack. Those avenues which are apparent require very specific, and unrealistic assumptions on the system [9].

B. Strengths of Provable Security

A security proof, by design, is infallible. This means that the design itself, if truly a proof, cannot be wrong. When a cryptosystem has every assumption clearly defined, the mathematics are sound and validated, the implementation model perfectly matches the proof model, and the proof model accounts for every possible attack vector, then the system should be confidently secure. Now in practice, this is truly a lot to ask which will be discussed further in section II C. Weaknesses of Provable Security. In fact, the current methodology of provable security is great for increasing the merit of a piece of security software. Consumers should have a clear method to see the software or hardware they are purchasing is mathematically considered secure. If provable security did not exist, then it would be very difficult to communicate to the average consumer or client the effectiveness of a given security system. Instead, other metrics such as energy consumption, RAM allocation, and time to compute would be the most important factors, which clearly would favor having no cryptography method in place at all.

Over time, cryptography will continue to grow, develop, and change as technology requires it to. When quantum computing becomes accessible to adversaries, any cryptography proof which is only computationally secure would be more vulnerable. A strength of provable security is that if it is continually updated to the changing technology environment, the math at the base of the proof should go unchanged. What will change are the assumptions, and the limit of computations considered secure. One aspect to consider when creating a provably secure cryptosystem is if any assumptions will change over time. Often Proof "A" will rely on Proof "B's" security as a point of comparison. This is to say, if the community has accepted the security risk of Proof "B" and Proof "A" is equal to or more secure than Proof "B", then Proof "A" should be considered safe to adopt. This decision to mark a proof as secure or not secure appears to be a binary decision, "provably secure" or "not provably secure". Then the system is compared to other systems at other bit-levels to "score" its complexity. The goal of one-way equations for example is to implement a calculation with low-computational costs in one direction, and massive computational costs in the other direction. Additionally, if this can be done with minimal assumptions, then the system is more likely to be implemented in a range of software environments.

C. Weaknesses of Provable Security

The purpose of listing cyber security proofs is to clarify the use cases and the security guidelines for which that protocol should operate. Additionally, one of the goals of cybersecurity and proving cryptographic techniques is to locate vulnerabilities in a system and patch or replace them where possible. From evaluation of the general belief across the field of cryptography, it appears most experts believe that mathematical proofs of rigorous algorithms will guarantee the security of a system. Yet, in examples which will be expressed later in this paper, occasionally a provably secure system will later be attacked. The reason for this occurrence is flaws in the initial proof, due to human-introduced errors. Faulty assumptions, implementations, or logic can lead to a system which was intended to be provably secure, but in practice is not. This could be the fault of the researcher who published incorrect data, or the consumer who is implementing the final software. Regardless, provable security is a very marketable term, and if it is slapped on every system with cryptography it begins to lose its meaning if no one is willing to validate if the claims are accurate. Additionally, it implies that it cannot be improved when that is in fact not the intent.

According to Neal Koblitz and Alfred Menezes, whom notably appear to be ostracized for their opinions on this subject, state in their paper Another Look at Provable Security "However, in our view the theorem-proof paradigm of pure mathematics is of limited use in evaluating the real-world security of a cryptographic protocol, and the very term 'provable security' is misleading. Over the years many problems have arisen with the way proofs are used in cryptography" [10]. Now although Koblitz and Menezes are not venerated amongst cybersecurity professionals, their purpose is justified here. In order to ensure a proof is productive towards the wider field of cybersecurity and cryptography, rather than something which will later need to be withdrawn, some portion of the efforts of cryptographic research must go towards validating the security of the proofs done by others. In this same document, these re-

searchers list several reasons why they believe security proofs could have human-introduced flaws. Some of their reasons will be discussed in detail as, even though these authors are not very popular among cryptography, their thoughts encourage the scientific process of testing and retesting. It is a safer assumption to be skeptical on the validity of information going when verifying the integrity of the system, because it takes a single false assumption or misconception to bring down an entire paper. When a security proof is published, peers are expected to validate the proof; however, this requires a level of critical thinking which is difficult to measure. In the same way no one in a crowd of people will come to the aid of another person, it's possible that every other researcher will assume someone else is more qualified or capable of assisting the research.

The first concept these researchers discussed as a reason for why provable security is not a great metric for the strength of a cryptosystem is because of the sociological climate of the research profession [10]. Often researchers are pressured to publish frequently which means there are a lot of papers being produced in less time. This phenomenon causes rushed work due to imposed deadlines and less availability for professionals to judge the increasing volume of research published each week. If these deadlines were extended, it would be expected that the quality of each publication would increase and the quantity would decrease. This creates a formula for producing less publications with less faulty assumptions. Additionally, if researchers are under stress, they are more likely to make mistakes or make less ethical decisions. For example, current sociology appears to encourage ignoring a flaw in research. If a researcher spends months on a paper and when proof-reading before the final publication notice a minor flaw, is it more beneficial for them to publish it and hope that flaw goes unnoticed or is it better for them to admit to their publisher that the research will need to be thrown out or delayed as it has inherent flaws. The next flaw they have pointed out is in regards to standardization of definitions for use in cryptography. Koblitz and Menezes claim that the definition of a model used in a proof may be different than the definition used in practice and that assumption alone may create implementation issues. Another method discussed is how side-channel attacks are often ignored in provable security as it is impractical and/or improbable to model these in a proof [10]. Adversaries are not going to make that same assumption, which creates an avenue for attack which is not covered by the proof. The last consideration when validating the integrity of a cryptosystem is if the proof can function as a cryptosystem. Koblitz and Menezes consider an example where known proofs are used to construct a protocol; however, any features which are normally present in the protocol are stripped if they are not needed to support the proof [10]. In this case, a valid proof is present which may not perform all the necessary actions of a cryptosystem which allows for attack vectors to exist, while maintaining the security of the proof. If a cryptosystem can both be provably secure, in that the proof provided in the system is valid, and have flaws which are not accounted for in the parameters of the proof, then should the cryptosystem continue to hold the title of "provably secure"? Should "provable security" consider zero-day scenarios in which the proof's developer had not considered or should it only consider known and standard methods of attack, such as by brute force?

This paper is not to argue that a rigorous, mathematical proof is not enough security for a cryptosystem. Instead, this paper hopes to bring light to faults in the manner provable security is implemented. Implementation is the key issue at play for provable security and security proofs. If a single assumption is fallacious, in that it makes a single wrong assumption, then the entire proof can be thrown along with any other mechanisms which rely on that mechanic as a building block. The argument comes down to a basic concept. Practical application of a cryptosystem is far more complex and has far more factors than a mathematical proof can account for. While that proof alone may be sound, coining that proof 'provably secure' can imply a level of security it cannot support in practice due to implementation concerns. Additionally, cryptosystems can gain the title provably secure after a proof is developed; however, only lose the claim to that title if the proof is broken. This created an "implied yes" system where a lack of effort allows a weakness to exist, rather than an "implied no" system where the absence of effort will deny the system to exist. An "implied no" system in this context cannot be applied to a proof as the definition of a proof in science does not need justification whereas a theory needs to be tested and applied repeatedly to confirm its validity. Theories in science cannot be proven; however, a mathematical proof is designed to be true without requiring testing. The issue is a proof relies on each of its assumptions to be maintained and often one or more of these can be altered by an attacker or through the implementation of the system which compromises the proof.

III. Examples of Provably Secure Systems, with Known Flaws

An analysis of six protocols guarded by provably secure cryptosystems is conducted. These protocols are commonly used every day for different use-cases of internet access, encrypting sensitive credentials, and transaction verification. Initially thought to be airtight protocols, numerous implementation errors revealed minor mistakes in proofs resulting in insufficient security which can be exploited by malicious users and eavesdroppers. These protocols are examined by their appropriate utilization, their structure of security, any problematic attacks discovered, and if applicable, a solution proposed by researchers.

A. Long Term Evolution

Long Term Evolution (LTE) is a mobile communication standard that is cost-effective for most Internet of Things (IoT) applications. LTE plays a key role in the information society by combining strict performance goals with modern security systems. It serves both casual and mission-critical

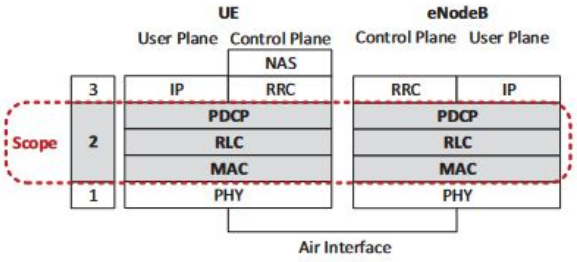

Fig. 2. The red area marked as scope defines the layer the data manipulation attack which defied the provable security of LTE can occur on.

infrastructure as well as public safety uses. These scenarios are demanding towards a secure specification of LTE.

The LTE protocol stack consists of the physical layer, the data link layer, and then the network layer. The data link layer, layer two, is responsible for providing three protocols: Medium Access Control (MAC), Radio Link Control (RLC), and Packet Data Convergence Protocol (PDCP). Each of these layers can be visualized in Figure 2. Previous work on the LTE protocol has identified crucial attack vectors for both layer 1 (physical) and layer 3 (network).

An in-depth analysis of layer two reveals three attack vectors which impair security and confidentiality. Identity mapping and website fingerprinting can exploit metadata in two similar passive attacks. Rupprecht et al. use the example of a website fingerprinting attack, which can exploit LTE's resource allocation. The other attack is an active attack which enables DNS redirection. This results in a DNS spoofing attack. All three attacks can be performed in realistic scenarios as the assumptions required for these attacks to occur are common mistakes seen in the implementation and use of the LTE technology.

When discussing the data manipulation attack, Rupprecht et al. stated "Even though the LTE Authentication and Key Agreement (AKA) is formally proven secure, this attack is still possible due to the lack of integrity protection of user plane data" [11]. This was not considered in the security proof, which explains how this was later found. Rupprecht et al. created a malicious relay which allows the usage of a chosen-ciphertext attack which can be conducted in a realistic scenario (and was not considered in the original set of assumptions). The resulting data shows that it is possible to create "DNS requests and spoof the DNS responses" [11]. Although no solution is proposed, the authors call for a solution to be put in place to amend the gap in the proof and to counteract these problems such as LTE's missing user data integrity protection.

B. Onion Routing

Onion routing is commonly utilized as a technique to anonymously communicate over a network. An onion network encapsulates the messages in several layers of encryption. As opposed to normal GET requests to a server, the connection is sustained through hopping between several servers, known as nodes, until reaching a destination server, in which the message is sent back through the same path of nodes.

To ensure safe and secure onion packet delivery, the Sphinx format simplifies building protocols for Onion Routing while HORNET focuses on improved performance through low latency by using Sphinx. Sphinx is a provably secure mix-format which means it allows clients to communicate information through mix servers. Kuhn et al. uncovered an attack during HORNET's "data transmission phase that allows it to link senders to receivers and large parts of the messages to their senders as well" [6].

A vulnerability was recently found that removes anonymity in HORNET, breaking the once thought secure system. The original Onion-Security property was thought to prevent the attack on HORNET, but this property was not correctly proven, creating this exploit in practice. Kuhn et al. point out two mistakes in the proof:

- First, there are insufficient properties that are meant to imply privacy.
- Second, only one of the properties is missing an important piece, another is inexact, and the last two do not provide any additional privacy.

Kuhn et al. created two additional properties to imply more secure privacy "Tail-Indistinguishability" and "Layer-Unlikability" [6]. A second mistake was revealed with consideration to their attack on HORNET: The properties known as "Camenisch" and "Lysyanskaya" were not correctly proven for the protocols. Kuhn et al. propose new properties that prevent well-known attacks on Onion Routing as long as they can comply with ideal functionality's adversary mode [6].

The adapted Sphinx proposed by Beato et al. should be able to solve the DDH problem. This repairs the anonymization of delivery for the encrypted message by introducing integrity checks at each hop to ensure no data or metadata has changed. Kuhn et al. note that the proof of this new adapted Sphinx is comparable to the original, where the symmetric key si is a secret, "no dependencies between FormOnion", "no modification", "no linking", and "Onion layer indistinguishable from random ones" [6]. The paper by Kuhn et al. goes on to state "The existence of this attack in HORNET and the improved Minx however contradicts their security proofs. The reason for this is not the flaws in the properties of [references "A formal treatment of onion routing" by Camenisch and Lysyanskaya], but rather, a common mistake in proving the properties" [6]. The flaw in the proof was simply a mistake where the layer which verifies external sources, known as an oracle, was not considered in the proof. This extra addition is not secure and introduces a vulnerability the proof did not consider. It is unclear whether this was left out of the formal proof because it made Onion Routing appear more secure, or if it was an honest mistake to exclude this portion.

C. Wi-Fi Protected Access II

Wi-Fi Protected Access (WPA2) is an encryption protocol designed to secure wireless computer networks. It focuses on the data link layer, IEEE 802.11. WPA2 became the standard

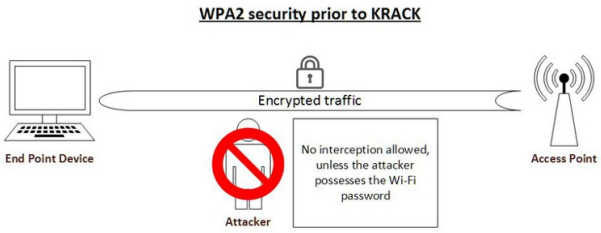

Fig. 3. WPA2 was proven to be secure; this figure shows how it was believed to function prior to the KRACk attack being discovered.

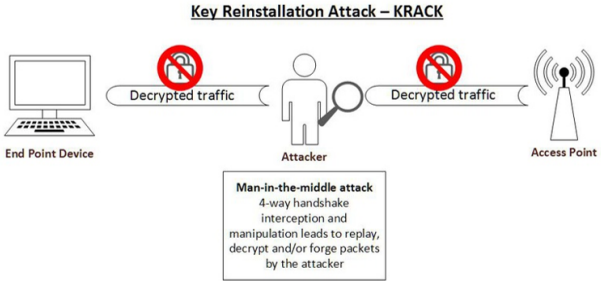

Fig. 4. WPA2 is no longer considered secure as a Key Reinstllation Attack is possible due to a weakness discovered in 4-Way Handshake communication method.

and ratified in 2004 by IEEE, was previously thought to be secure until KRACK attacks [12]. See Figure 3 for how WPA2 was thought to work before KRACK attacks were discovered.

First, a replay attack was discovered in 2016 by Frank Piessens and Dr. Mathy Vanhoef. This attack is called the KRACK attack, which stands for Key Reinstallation Attack. This works by exploiting a reset nonce vulnerability to steal data over these wireless networks due to a vulnerability in the third step of WPA2's Four-Way Handshake. See Figures 4 and 5 for how and where the Key Reinstallation attack occurs within the 4-Way Handshake [13].

Dr. Vanhoef notes that the victim re-uses nonces when encrypting the data frames, allowing attackers to recover encrypted data. An implementation bug discovered in certain operating systems such as Linux and Android results in the failure to reinstall the secret key, rather than an all-zero encryption key. This results in a trivial process of intercepting and manipulating all the transmitted data from these Wi-Fi-utilizing devices [12]. The chart in Figure 6 shows which devices are vulnerable to different handshake attacks and portions of the handshake attacks. If a column lists N/A it is because those steps cannot occur because a necessary aspect of the vulnerability is blocked.

Hue et al. details an "Evil Twin attack", a man in the middle attack used to snag a user's single sign-on (SSO) credentials through setting up a malicious access point [14]. Then, Hue et al. conducted an analysis from "7275 configuration instructions from 2061 TEIs" which mostly contained insecure setups, resulting in an 86% rate of TEI's falling victim to

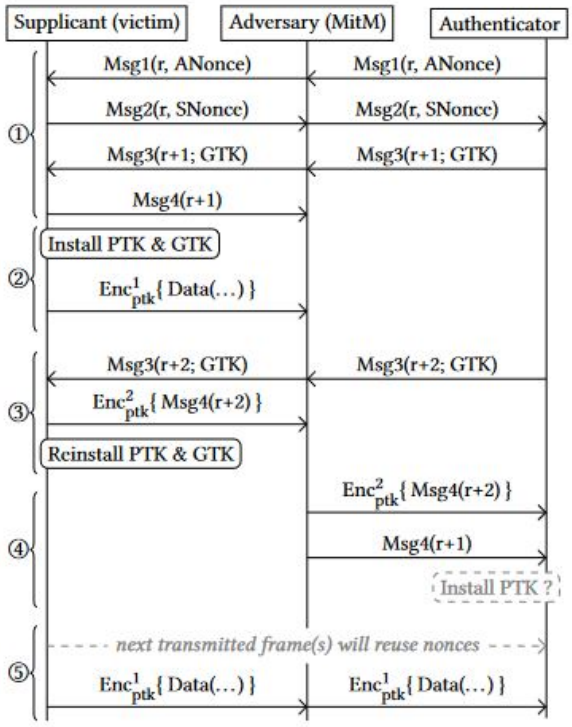

Fig. 5. If the victim accepts plaintext transmissions of the third message, a new session key can be installed.

Implementation	Re. Msg3	Pt. EAPOL	Quick Pt.	Quick Ct.	4-way	Group
OS X 10.9.5	✓	✗	✗	✓	✓	✓
macOS Sierra 10.12	✓	✗	✗	✓	✓	✓
iOS 10.3.1 [c]	✗	N/A	N/A	N/A	✗	✓
wpa_supplicant v2.3	✓	✓	✓	✓	✓	✓
wpa_supplicant v2.4-5	✓	✓	✓	✓[a]	✓[a]	✓
wpa_supplicant v2.6	✓	✓	✓	✓[b]	✓[b]	✓
Android 6.0.1	✓	✗	✓	✓[a]	✓[a]	✓
OpenBSD 6.1 (rum)	✓	✗	✗	✗	✗	✓
OpenBSD 6.1 (iwn)	✓	✗	✗	✓	✓	✓
Windows 7 [c]	✗	N/A	N/A	N/A	✗	✓
Windows 10 [c]	✗	N/A	N/A	N/A	✗	✓
MediaTek	✓	✓	✓	✓	✓	✓

[a] Due to a bug, an all-zero TK will be installed, see Section 6.3.
[b] Only the group key is reinstalled in the 4-way handshake.
[c] Certain tests are irrelevant (not applicable) because the implementation does not accept retransmissions of message 3.

Fig. 6. Each column asks the following questions: if the transmission is accepted (column 2), if plaintext messages are accepted when a session key is configured (column 3), if messages are accepted in sequence / false if delay is necessary (column 4), if vulnerable when clients only allow encrypted messages when a session key is configured (column 5), if the 4-Way handshake is vulnerable (column 6), and if the group key handshake is vulnerable (column 7).

credential theft [14]. In order to perform an Evil Twin attack, an adversary simply needs to route DHCP through a soft Access Point using IP Tables. Then the adversary can de-authenticate a target WPA2 router, which will disconnect its users from its network. When the victim goes to reconnect, they may accidentally choose the Evil Twin access point which still works to provide internet; however, all of the traffic can be collected and monitored by the adversary.

Security patches have since been released for operating systems on several devices including but not limited to:

- Windows
- Linux
- macOS
- Chrome OS computers
- iOS phones
- Android phones
- several other distributed systems such as those used in printers, fridges, etc..

To help mitigate the chances of an intrusion, when the key is being installed a client can disable the EAPOL-Key. This will deny attackers access to the network as long as TDLS is disabled. However, this method is not always feasible due to reduced network connectivity speeds with this setting selected. Network speeds are adversely affected by this method as every time a packet is lost, rather than resending the packets with the same nonce, a new nonce is generated and the packets are resent after a delay of a couple seconds. Another method of mitigating the chances of intrusion is to avoid wireless connections if one is unsure as to the status of the security patches.

D. Wi-Fi Protected Access III

The WPA2 protocol was superseded by WPA3, which was released on July 1, 2020. WPA3 uses 192-bit cryptographic strength and replaces the pre-shared key (PSK) with a Simultaneous Authentication of Equals (SAE) protocol which is again analyzed by Dr. Vanhoef [12]. This is intended to provide a more robust password-based authentication used for Wi-Fi passwords. These under the hood changes, known as Dragonfly (seen in Figure 7), provide forward secrecy and protect against offline dictionary attacks.

Although this protocol implements the Dragonfly handshake, the promise of guaranteed security is not fulfilled due to unclear guidelines. Vanhoef and Ronen go into detail, stating "a close variant of Dragonfly received significant criticism while being standardized, while a different variant of Dragonfly has a formal security proof. These contradictory claims raise the question of whether Dragonfly is secure in practice" [12].

The Dragonblood attack is a collection of attacks against WPA3, utilizing password dictionary attacks. These attacks abuse cache-based leaks and allow attackers to recover the Wi-Fi password through its encoding method [12]. When these attacks occur, victims are exposed to password partitioning attacks through these design flaws.

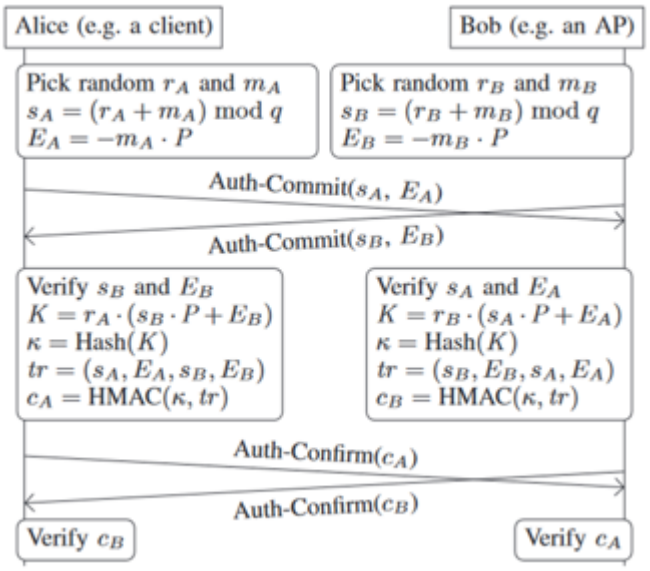

Fig. 7. Dragonfly Protocol for WPA3 illustrated by Vanhoef and Ronen.

Vanhoef and Ronen state "for instance, searching through a dictionary of size 10^{10}, which is larger than all available password dumps, can be done for less than $1 in GPU-enabled Amazon EC2 instance" [12]. An attacker is able to pay for the service of another computer to run a dictionary attack for only a few cents. Attacks which are so accessible can be performed by a larger population and are considerably more impactful due to the rate at which they can possibly occur.

Vanhoef and Ronen also provide several tools to reproduce their analysis and results in research. Dragonforce, for example, uses a password partitioning attack, Dragontime attacks the SAE handshake, Dragondrain tests the denial-of-service attack vulnerability for the SAE handshake, and Dragonslayer which "performs invalid curve attacks against EAP-PWD clients and servers. These attacks bypass authentication: an adversary only needs to possess a valid username" [12]. Each of these tools used within their analysis only require a few pieces of knowledge and assumptions which are realistically attainable and should be considered a major weakness.

To patch the protocol and prevent these attacks, only minor adjustments to the protocol and handshake are required. Vanhoef and Ronen state "we believe more openness while creating WPA3 and Dragonfly could have prevented most attacks. For example, excluding the MAC addresses from Dragonfly's password encoding method, or using constant-time algorithms, would have mitigated most attacks" [12]. For reference, WPA3 was mostly created behind closed doors and did not utilize much peer review before its announcement.

Besides security, one current WPA3 issue is ensuring compatibility; not all Wi-Fi-supported devices can recognize WPA3 due to hardware and software limitations since it has a more recent release date and requires specific hardware specifications.

WPA3 may provide a newer, modern scheme but suffers

from faults which can be exploited using a collection of Dragonblood attacks. The authors propose minor changes in the protocol which mitigate most of the attacks, such as moving the MAC address of the peer to later stages of the handshake and excluding it from the password encoding algorithm. Another improvement shows that the random token for the server can be excluded for EAP-PWD [12].

E. Multi-Signature Protocols

MuSig (Multi-Signature) is a relatively recent protocol that replaces other models such as the Elliptic Curve Digital Signature Algorithm (ECDSA). It is a Schnorr-based multi-signature scheme that is provably secure. As this scheme allows for key aggregation, "meaning that the public keys of all cosigners can be aggregated into a single public key *pk*. As a result, verifiers do not need to be given the explicit list of all participants' public keys anymore and can just use the aggregate key instead" states Nick et al. [15]. Only the aggregate key is used by the verifiers, enhancing the signer's privacy, and allowing for modifications of the multi-signature protocol as needed.

Assuming a situation where an attacker convinces a user to produce two distinct partial signatures which exposes their private key due to the signatures having the same randomness, then this attack is possible. This would be possible but improbable; however, was not discovered in the original security proof. Nick et al. state that currently there is no multi-signature scheme based on Schnorr signatures which is able to be implemented without knowing the state at signing time or having access to the secure randomness (such as a seed). Further robustness is required with a revision to MuSig; Nick et al. suggested to have signers deterministically generate their nonce. In addition, the previously required three rounds are reduced to two in a single signing session required by Schnorr multi-signatures.

MuSig-DN is a revised variant of the MuSig Scheme, also known as MuSig with Deterministic Nonces, which "allows signers to generate nonces deterministically and without having to maintain state [15]. A pseudorandom function, F with secret key u_i (the nonce key), is applied to participants' public keys and the encrypted message. Their public nonce, Ri = r_i G, is sent with a non-interactive zero-knowledge (NIZK) proof which is examined by the cosigners through a public key called the host key, U_i, which can ascertain that a false signer will be caught by the other participants [15]. This known piece of information allows the true signers to identify the false signer who will not know the NIZK information.

The authors believe that these sets of techniques are useful beyond the field of deterministic multi-signatures. A common use-case scenario for MuSig is in the cryptocurrency space, where verifiable encryption takes place to a certain escrow agent trusted by a good's buyer and seller. When paying in Bitcoin, the buyer contains a verification key of $A = aG$ and does not trust the seller with verification key $B = bG$ [15]. The currency is sent to a jointly controlled verification key X, generated from A and B in MuSig. Secret key "a" is encrypted using VES and delivered to the escrow agent, where the escrow agent is not involved in the transaction, but still obtains "a" through VES [15]. The authors implement their technique through the secp256k1 elliptic curve which is the ECC scheme used for Bitcoin's public key cryptography. Although more complicated to implement, MuSig-DN is not vulnerable to state attacks such as rewinding and repeated session attacks, making it the more preferable alternative for the verification of secure cryptocurrency transactions.

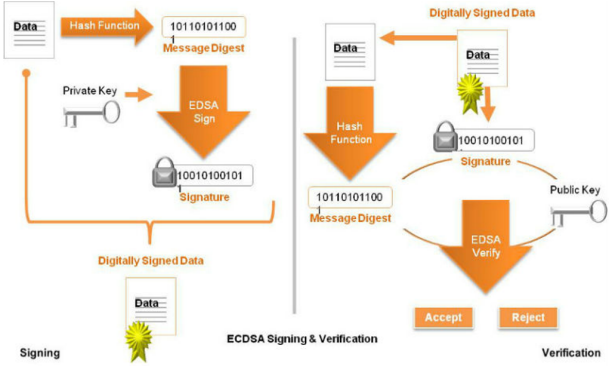

Fig. 8. The above diagram shows how the signing and verification models are intended to function within ECDSA.

F. Elliptic Curve Digital Signature Algorithm

The Elliptic Curve Digital Signature Algorithm (ECDSA) combines the benefits of both Elliptic Curve Cryptography (ECC) and the Digital Signature Algorithm (DSA) and Figure 8 shows how signing and verification occurs under this model [16]. ECDSA produces shorter private key lengths than RSA, however, this method maintains and, in some cases, improves the overall security. Regardless, ECDSA is not without vulnerabilities derived from implementation flaws and side-channel vulnerabilities. Overall, the majority of attacks have used at least two bits of nonce bias, with the outlier being 80-bit security level curves containing 1-bit biases [5]. A nonce is a sensitive random value which is privately distributed over an interval of integers. In Figure 9, the nonce is sent by the Sender in Step 3 to act as a challenge [17].

The Montgomery ladder is an algorithm of computing scalar multiples of points across a broad class of elliptic curves. Aranha et al. present a class of exploitations known as Ladderleak against the Montgomery ladder which elusively reveals a single-bit of the high-probability secret scalar, corresponding to the nonces [5]. OpenSSL is the use-case for this attack, taking advantage of both prime and binary curves. This is considered a side-channel attack and is effective at revealing the full private key of an ECDSA system at a low cost.

OpenSSL runs a popular scalar multiplication algorithm known as López-Dahab, which runs coordinates x = X / Z, $y = Y/Z^2$. This algorithm exposes certain cache-timing vulnerabilities to adversaries in both binary and prime curve cases. The bit is retrieved from Flash+Reload (FR) cache timing attacks in the side-channel. This stems from an issue

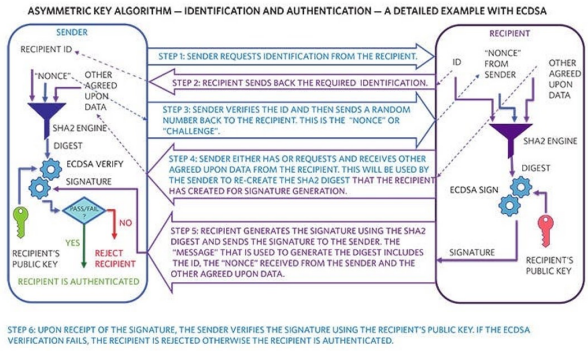

Fig. 9. This diagram shows the communication between an untrusted recipient and a sender. The diagram uses purple to show Recipient communications and a blue-green color to communicate Sender communications.

within the OpenSSL library itself, revealing the second most-significant-bit of the scalar by allowing the adversary to launch the cache-timing attack [5]. Aranha et al. implemented these attacks using a "FR-trace program in the Mastik side-channel analysis toolkit" [5].

The Bleichenbacher's attack is a set of adaptive, chosen-ciphertext attacks where the adversary knows the sequence of bytes from the encryption message's length. These attacks primarily:

- Find a vulnerability in the oracle, the layer which verifies external sources,
- Uses a Fourier analysis-based approach,
- Crucially rely on small and sparse linear combination complexities.

Aranha et al. implemented this attack from provided source code. Despite being one of the most widely-used signature schemes in modern encryption, ECDSA has a number of flaws from implementation, particularly due to the nonce [5]. The authors recommend the following countermeasures aimed towards preventing the LadderLeak attack [5]:

- Enforcing constant-time behavior in the implementation of scalar multiplication
- Z-coordinate randomization
- Alternative scalar multiplication algorithms

The simplest of the three countermeasures to implement is the randomization of the Z coordinates. It is a popular countermeasure when implementing the Montgomery ladder algorithm for elliptic curve 25519, however its efficacy is currently uncertain in regards to countering side channel attacks [5].Therefore caution must be always exercised when utilizing this countermeasure. Enforcing constant-time behavior in the implementation is another satisfactory option, however this can cause a performance impact, as noted by the authors [5]. Furthermore, the authors go on to note that there are other scalar multiplication algorithms that do not affect performance to the same extent and are perhaps simpler to implement in a secure manner [5].

IV. Conclusion / Discussion and Outlook

To extinguish any possible threat of eavesdroppers or other security exposures, channels of communication often are secured via NP-hard problems such as in WPA3 with the Dragonfly handshake, Onion Routing, and Long-Term Evolution. Security proofs are the foundation of these implemented cryptosystems, given underlying assumptions such as Learning with Errors, the Discrete Logarithm problem, integer factorization, and multilinear maps.

After analysis of each implemented protocol, their embedded provably-secure cryptosystems, and the theoretical and implementation flaws, several ideas can be learned to apply with future cryptosystems and their implementations. First, further validation to cryptosystem mathematical proofs should be applied. This should reduce error at the theoretical level, with examples seen such as HORNET's two properties' mistakes in Onion Routing. Developers should also analyze each layer relevant to the cryptosystem before implementation, as seen with the exploits in LTE due to an under-analyzed data link layer and Ladderleak for ECDSA's Montgomery ladder exploits. As seen with the Dragonblood attacks in WPA3, minor holes in the cryptosystems can result in several exploits. Subsequently, given cooperation between developers of the software and cryptosystems, these vulnerabilities can be easily patched, as seen with MuSig and MuSig-DN.

Furthermore, software testing techniques to these protocols that use provably secure cryptosystems should be applied. Black box testing such as boundary value analysis, equivalence class testing, error guessing, decision table testing, and non-functional testing techniques such as performance testing.. Performance testing includes load testing, stress testing, spike testing, volume testing, endurance testing, and scalability testing. White box testing should also be applied in the code itself, such as execution testing, mutation testing, and integration testing especially due to these cryptosystems which can be seen as extensions of the software. These testing techniques should be able to detect implementation bugs that remove security in these protocols and reduce the chance of bugs resulting in security vulnerabilities.

Mathematics is confined and valid when every assumption in a cryptosystem is properly satisfied, both in practical and theoretical application. It has been shown that the property of being provably secure does not imply airtight security upon implementation. Hard problems exist within the realm of provable security, serving as the technical standard for safeguarding against brute-force attacks. Mathematical fallacies in proofs are also to blame for protocol failures, such as in the Onion Routing proof where the oracle is not considered. These mistakes in the theoretical space with mathematical proofs and the practical space via implementation create opportunities for bad actors to exploit vulnerabilities.

Given the state of these broken and improved cryptosystems, there must be a call to prepare for the quantum-computing era. Academia needs researchers such as Dr. Vanhoef to continue to stay and devise algorithms and tools to verify the accuracy

of the underlying proofs of provably secure cryptosystems. Research progress shows how minor adjustments to protocols such as WPA3 can prevent risks of cyberattacks and open new risk avenues. Professional associations should also steepen their requirements for cryptosystem proof standards requiring examples and airtight logic to reduce the risk of human error. The stateless signing benefit of MuSig-DN offers an appropriate trade-off of high computational cost given a world approaching quantum computing. Additionally, the security-bit comparison from Table 1 should be the more public-facing standard as it shows relative strength of each system, and included symmetric keys cryptography such as AES. In addition, open source verification and vetting of a proposed cryptosystem or protocol should be the standard, when possible. Many vulnerabilities, such as the vulnerabilities discovered in WPA3, would have likely been found before distribution of the protocol if an open source, or at least a broader, vetting had been undertaken.

REFERENCES

[1] E. Barker, "Recommendation for key management - NIST," NIST.gov, May-2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf. [Accessed: 06-Dec-2021].

[2] I. Lotfi, "(PDF) Cryptographie à Base de Courbes Elliptiques," ResearchGate, Jun-2017. [Online]. Available: https://www.researchgate.net/publication/323946296_Cryptographie_a_base_de_courbes_elliptiques. [Accessed: 07-Dec-2021].

[3] V. Klima, "Tunnels in hash functions: MD5 collisions within a Minute," Cryptology ePrint Archive, 01-Mar-2006. [Online]. Available: https://ia.cr/2006/105. [Accessed: 06-Dec-2021].

[4] NIST-CSRC, "Update to current use and deprecation of TDEA," CSRC, 11-Jul-2017. [Online]. Available: https://csrc.nist.gov/news/2017/update-to-current-use-and-deprecation-of-tdea. [Accessed: 06-Dec-2021].

[5] Aranha, D. F., Novaes, F. R., Takahashi, A., Tibouchi, M., & Yarom, Y. (2020, October). Ladderleak: Breaking ecdsa with less than one bit of nonce leakage. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 225-242).

[6] Kuhn, C., Beck, M., & Strufe, T. (2020, May). Breaking and (partially) fixing provably secure onion routing. In 2020 IEEE Symposium on Security and Privacy (SP) (pp. 168-185). IEEE.

[7] A. W. Dent, "A Brief History of Provably-Secure Public-Key Encryption," ePrint IACR, 2009. [Online]. Available: https://eprint.iacr.org/2009/090.pdf. [Accessed: 06-Dec-2021].

[8] NIST, "Cryptographic competitions," Crypto competitions: AES: the Advanced Encryption Standard, Jan-2014. [Online]. Available: https://competitions.cr.yp.to/aes.html. [Accessed: 06-Dec-2021].

[9] S. Li, "Symmetric key algorithm," Symmetric Key Algorithm - an overview — ScienceDirect Topics, 2017. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/symmetric-key-algorithm. [Accessed: 06-Dec-2021].

[10] N. Koblitz, "Another Look at 'Provable Security,'" ePrint IACR, Jul-2004. [Online]. Available: https://eprint.iacr.org/2004/152.pdf. [Accessed: 06-Dec-2021].

[11] D. Rupprecht, K. Kohls, T. Holz and C. Pöpper, "Breaking LTE on Layer Two," 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 1121-1136, doi: 10.1109/SP.2019.00006.

[12] Vanhoef, M., & Ronen, E. (2020, May). Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In 2020 IEEE Symposium on Security and Privacy (SP) (pp. 517-533). IEEE.

[13] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," MathyVanhoef Papers, 2017. [Online]. Available: https://papers.mathyvanhoef.com/ccs2017.pdf. [Accessed: 06-Dec-2021].

[14] Hue, M. H., Debnath, J., Leung, K. M., Li, L., Minaei, M., Mazhar, M. H., ... & Chau, S. Y. (2021, November). All your Credentials are Belong to Us: On Insecure WPA2-Enterprise Configurations. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 1100-1117).

[15] Nick, J., Ruffing, T., Seurin, Y., & Wuille, P. (2020, October). Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 1717-1731).

[16] J. Finney, "Understanding blockchain: Security, Risks & Auditing Tips," Linford & Company LLP, 16-Jul-2021. [Online]. Available: https://linfordco.com/blog/blockchain-security-risks-auditing/. [Accessed: 06-Dec-2021].

[17] Z. Sardar, "Cryptographic fundamentals — electronic design," Electronic Design, Apr-2020. [Online]. Available: https://www.electronicdesign.com/technologies/embedded-revolution/article/21128993/maxim-integrated-cryptographic-fundamentals. [Accessed: 06-Dec-2021].

The citations which are bolded represent six citations selected from appropriate sources in 2019 - 2021 per the deliverable requests in CSCI 55500 Cryptography.