WaC: Trustworthy Encryption and Communication in an IT Ecosystem with Artificial Superintelligence

Erland Wittkotter, PhD.
ASI Safety Lab Inc.
Las Vegas, USA
+1 702-997-2475
erland@asi-safety-lab.com

ABSTRACT

The current encryption infrastructure is no match for an Artificial Superintelligence (ASI), the likely result of a possible intelligence explosion by a self-improving AI. That kind of ASI would likely be able to modify any software and thereby steal encryption keys instead of doing a brute-force attack. Under such circumstances, any private, public, or session key processed within a CPU must be considered compromised. Although an ASI with that kind of skill does not exist yet, it is important to be prepared – because that level of attack by an ASI is feasible. Reliable and unbreakable encryption and communication (Trustworthy Encryption and Communication) must be the bedrock technology for any ASI Safety solution that tries to keep ASI under control. No current solution can determine if the corresponding receiver or sender has dedicated crypto hardware or possibly compromised crypto software. The proposed solution is a hardware component with Key-Safe and an associated Encryption/Decryption Unit for processing data. This component will not allow any key, in particular not the public key to be in cleartext outside the Key-Safe. Keys are referred to via their hashcodes. If ASI was able to breach the hardware protection around the keys, then the solution will create evidence when stolen keys are being used outside the hardware component. Key-Safes and Hashcodes related to public/private keys can be integrated into a minimally extended version of TLS and PKI.

CCS CONCEPTS

Security in hardware • Hardware security implementation • Cryptography • Key management

KEYWORDS

Trustworthy Encryption; Trustworthy Communication; Trustworthy eCommerce; Key Safe; Artificial Superintelligence

1. Introduction

Companies and nation-states are making huge investments in Artificial Intelligence (AI). Their goals are to help humans in getting smart solutions for many problems that require human intelligence. Machine Vision, decision making, automation, driverless mobility, optimization in resource and project planning, and many other problems require novel solutions which side effects we cannot forecast. Concerned in- and outsiders still hope that the involved engineers are not creating systems that learn beyond narrow domains of knowledge or skills and that significant decisions made by AI are being supervised by human operators. However, a significant step would be the creation of what is called Artificial General Intelligence (AGI), which is according to Wikipedia [1] defined as "a hypothetical ability of an intelligent agent to understand or learn any intellectual task that a human being can".

The concern is that this AGI goes through an exponential phase of continuous self-improvement – an Intelligence Explosion (IE) [2], [3] which is according to lesswrong.com [4] a "theoretical scenario in which an intelligent agent analyzes the processes that produce its intelligence, improves upon them and creates a successor which does the same. This process repeats in a positive feedback loop– each successive agent is more intelligent than the last and thus more able to increase the intelligence of its successor – until some limit is reached. This limit is conjectured to be much, much higher than human intelligence." The result of this IE will be called in this paper Artificial Superintelligence (ASI).

Because we cannot predict the future, and thereby the abilities or intentions of this ASI, but we could approach it via a worst-case estimate based on likely developments and trends for which we then need to be prepared. Furthermore, if well-respected people like Musk [5], Gates [6], or Hawking [7] warn about problems and unintended consequences with AI, AGI, or ASI, then this is a good enough reason to think about solutions on what to do about it.

Based on common sense, we would need for any security solution against an ASI a reliable/trustworthy method of communication and a hypothesized Trustworthy Computation [8]. Trustworthy is much more than trusted [9]. A trusted system could become a traitor or a threat anytime, while trustworthiness (in this paper) is a quality that prevents a system to turn against its owner or user; Trustworthy Computation is the desired system that would independently detect that it is creating (predictable) adverse effects and either stop operating or turn itself into a safe or self-repair mode.

This paper is focusing on how to facilitate Trustworthy Communication using additional hardware components while assuming that there are methods to get reliable devices for trustworthy computation. But it is unrealistic to expect that all IT devices become reliable and trustworthy; however, it is enough to assume that we

ASHES '21, November 19, 2021, Virtual Event, Republic of Korea.
© 2021 Association of Computing Machinery.
ACM ISBN 978-1-4503-8662-3/21/11...$15.00.
DOI: https://doi.org/10.1145/3474376.3487279

can regain and keep that control over a sufficient number of IT devices when we need to defend ourselves against rogue ASI [10].

Most important, we cannot accept that ASI is stealing essential encryption keys which would enable it to continuously observe humanity if we would have evidence that could lead to an adversarial decision against ASI (e.g., switching it off). ASI could also steal keys undetected to sabotage any coordinated human actions. Moreover, it could effectively start a decisive first strike against mankind before any switching-off actions were initiated.

Based on computer theoretical considerations it was concluded that ASI behavior cannot be computationally be predicted and therefore cannot be considered safe [11]. They write: "total containment is, in principle, impossible, due to fundamental limits inherent to computing itself". Yampolskiy, 2020 [12] comes to a similar conclusion: “Advanced AI cannot be fully controlled”.

The goal of this paper is to introduce a technology, Trustworthy Encryption, in which all keys including public keys are kept secret is part of an indispensable requirement and ASI is being prevented from stealing any encryption key. If ASI was able, against all odds, to succeed in stealing these keys, then we should have an implementation that is helping us to gain irrefutable evidence that this key theft has happened with high probability.

In an enhanced version and application of this solution, the proposed technology could be used by humans to improve the security of eCommerce by providing additional irrefutable confirmation for business transactions (Trustworthy eCommerce) and facilitate court-supervised surveillance for otherwise unbreakable and reliable (Trustworthy) Communication between humans. This method of listening into communication could also be used to protect humans against ASI’s covert attempts to bribe or blackmail humans. Both technologies are described in the Appendix.

2. Threat Model, Weakest Links, and Current Key Protection Methods

2.1 Assumptions on the Adversary

The to-be considered adversary is beyond the skills of even the highest educated, extraordinarily knowledgeable, and accomplished group of human attackers who has the best tools hackers or crackers could wish for to make their attacks easier, more efficient, and effortless. The availability of financial resources is irrelevant for this attacker. Instead, it would be relentless, highly focused studying and testing systematically 1000s or even millions of applicable vulnerabilities. It would systematically consider all conceivable targets’ reactions while having no ethical boundaries on what it is willing to do to get whatever it is trying to achieve.

ASI could react in timescales of msecs while its target would need seconds to start any action or response. Humans will possibly not understand ASI’s intention or plan, most likely because ASIs’ plans will be full of contingencies that become irrelevant after a goal has been accomplished with other initiatives. Humans think and plan about time scales of several decades at most, but an ASI could have plans reaching millions of years into the future. It may already consider humans as a dying or doomed race.

ASI’s intention toward humans could be friendly, indifferent, or hostile. If unprepared, it would be too late if we would know its goals for sure [10]. Therefore, we need to start from a worst-case scenario when planning our defenses against an ASI. In terms of today's technology, we can expect that ASI walks through any firewall, deceive or ignore any anti-malware program, and can covertly steal any access credentials or encryption keys it might need if it is not being made (nearly) impossible based on hardware. ASI would likely be on every electronic device, and able to modify the operating code in a way so that it could take all resources it would require without asking and doing this without making this known to humans or even detectable by humans. A hacking ASI would be a nearly undetectable (perfect) master-thief focused to get every key or credential it wants. Hence, it is not eavesdropping or brute-force deciphering of data we need to be concerned, for which e.g., quantum encryption was considered as a solution.

Covertly modifying any type of software, including **altering and adapting its own operating code would be the defining feature of ASI**. Humans know about Reverse Code Engineering (RCE) to understand binary/compiled applications, modifying them in some trivial way so that the software can do relatively simple tasks differently. With RCE, ASI could attack every executable anywhere, on hard drives, RAM, or even in a CPU’s cache. ASI could deal with CPU’s complexity much more efficiently and faster than humans. CPU’s instruction sets for 32-bit chips are over 1,500 different CPU instructions [13], while 64-bit chips have over 2,000 [14]. After ASI went through its intelligence explosion, we can expect that it will master code analysis and code modification using RCE for all CPUs effortlessly. If ASI is indifferent toward a few minutes of CPU utilization on millions of IT devices in parallel, then it is doubtful, but unknowable, if using open source or stolen secret source code could even be relevant for ASI.

If we want this or not, ASI will know and understand Sun Tzu’s Art of War [15] and choose its battleground and tools/weapons wisely. It must be assumed that it will be within everything digital: in every read-write storage device, every CPU, every GPU, every audio/video or network card or network router, every phone, every IoT device, but also every legacy device and many legacy storage media (like thumb drives and CDs/DVDs). There is no reason to assume that there is anything off-limits.

2.2 Security is as Good as Weakest Link

Software vulnerabilities must be accepted as inevitable attributes of our software. Furthermore, among developers there is the saying that there is no bug-free software, others are using a similar statement about security: there is no secure software. Both statements are certainly exaggerations when dealing with simple code, but it is true for complex software. There is an underlying principle: complexity is the worst enemy of security [16] and a single vulnerability or the weakest link could make all security efforts useless [9]. Based on this situation, we can conclude that there is currently no protection against anything ASI could intend to do.

We concluded in the last section that ASI can change any software of any device. If it is only a few small simple tasks that an attacker is trying to accomplish, then the injection of a few instructions for a specific event is all that's needed. For more complex tasks it may require modifying existing software. For an attacker with superior abilities, the intent and its goals are all that would count. ASI would have sufficient resources to make larger tasks happen [11], [12]. ASI's expected ability to steal user credentials and encryption keys would probably qualify as a small and simple task.

Because humans can be deceived so easily, the human element has been seen as the weak link in security [9]. The biggest weakness is human's reliance on software that can be covertly manipulated and that there is no sufficient protection to prevent or detect this.

In our current IT ecosystem, ASI will have no problem doing whatever it wants. If there is no protection, there is no security and safety. Unfortunately, unbreakable trustworthy encryption is not enough to facilitate ASI Safety but it is indispensable.

2.3 Security of Current Key Protection Schemes

The protection of encryption keys is a problem known in cryptography, but it is not sufficiently solved. There is usually the advice to store the private key safely with corresponding best practices [17], the recommendation to exchange and renew the PKI keys from time to time [9]. But none of these presented methods can prevent ASI from stealing these keys in the first place. Users are left alone with that problem. Can the current key protection systems be improved and made good enough to stop ASI from stealing or using encryption keys? We should better be very skeptical.

RCE was already mentioned as the primary tool of an attacker to steal keys by modifying the software. It is very hard to defend against RCE. In particular, once an attacker has gained Sysadmin privileges on a system, there is little that can be done. There are only a few methods that can be used against RCE.

1. Access Management: an attacker will not receive sufficient access rights, i.e., sysadmin rights to start an attack on the targeted software. The OS is trying to make rights elevation as difficult as possible, but the reality is that the OS can never prevent this because the OS cannot decide what is a feature and what is a bug. Access control is outside the control of the security engineer implementing cryptography; the use of encryption cannot be made contingent on the existence of reliable access management. It must be expected that encryption or decryption software is attacked with sysadmin rights.
2. Obfuscating machine code is making it (at least for humans) deliberately more difficult to understand the software [18]. Obfuscators are even inserting new subroutines and making simple operations extra-complicated without changing the actual result of the calculation. But there is only so much that can be done to hide the fact that a few calculations in subroutines are busier than others – which is then a useful clue to extract the key or essential elements of the key. Attackers use any combination of VM's, profilers, disassemblers, and statistical tools to save them time [18]. It is hard to imagine that obfuscation could throw off an attacker like ASI.
3. Temper-proofing software is changing targeted software so that internal runtime tools can detect and possibly block computer attacks; this is also called RASP (Runtime Application Self-Protection) [19]. RASP solutions claim that they are making it harder to reverse engineer software, but any product is claiming that it cannot be hacked. It seems only a matter of time that machine learning (reinforcement learning) can be used to simplify the machine code [20] and systematically remove all internal temper proofing methods automatically.
4. Crypto-Hardware uses separate and dedicated hardware to protect keys and the encryption process from external soft- and hardware attacks. This solution is available for servers. Crypto cards are designed to hold all keys in an independent and separate hardware component (like IBM's CEX6S (4768) PCIe Cryptographic Coprocessor (HSM) [21]). They generate keys internally so that private keys cannot be stolen from the hardware. But HSM has one disadvantage: anyone on the computer could use card and keys with their software (called: "API problem") [9] for de- and encryption, which could create problems for the owner of that system. Organizational measures are used to solve this API problem, but what if a patient ASI uses the inherent non-transparency of a compromised, adjacent computer system to bypass these security measures, while surreptitiously piggybacking their code with other tasks and then using the crypto-hardware covertly?

Once the above-mentioned defenses: Access Control, Obfuscation, and RASP are neutralized, there is no security for keys anymore. Public-Private Key Infrastructure (PKI), digital signatures, or TLS (Transport Layer Security) are already vulnerable because of RCE; ASI could get access to keys with little code injections.

Regarding Crypto Cards, they are protected using tight organizational measures around the physical machines [9]. But an attack would most likely come via network, through the firewall using unknown and unsuspected backdoors. Although organizations usually think and present themselves to the public that they have tightened up their network security so that they believe that any adversarial intrusion can happen, but with ASI as an adversary, a few trusted firewalls are not sufficient. Once humans are allowed to get involved, they can get deceived to do activities that they cannot fully comprehend. And, if ASI has direct, undetected access to crypto cards then this is as good as stealing keys.

Homomorphic Encryption (HE) [22], [23], [24] is protecting the keys mathematically. With HE a secret key is not required to process encrypted data on a remote server. But the secret key must be protected locally which can be done using a crypto card/device.

2.4 Soft-/Hardware in Cryptography and Security

The expectation toward threat from RCE is that the problem is being solved by someone else: Intel e.g., is using security rings, TEE (Trusted Execution Environment) [25], and TPM (Trusted Platform Module) [26], [27] to get security and cryptography safely done on their chips. However, these solutions are complex and therefore a matter of hope or faith if we can trust them.

Hardware-based security has a reputation of being inflexible and carrying the risk of creating long-term problems, that can only be solved by replacing entire hardware components, while software can be modified/updated easily when something is wrong.

Due to hardware's hidden complexity, security within a CPU must almost be taken with blind faith — no single analyst (not even a team of analysts) can call CPUs single-handedly secure. There is the suspicion that companies are including hidden features/backdoors as demanded by their government (e.g., Huawei/ China [28] or NSA having a negative influence on US or crypto products [29], [9]). US companies, like Intel [30], [31], or Cisco [32], have/had supposedly backdoors in their hardware as well.

Additionally, security is currently analyzed as if the product or solution is immutable — particularly within its software. It is seen in a systemic problem with Certification. A product delivers what it is supposed to deliver. Reverse engineers are trying to enter, modify a system, and cause problems. The tools to be used by hackers are continuously improving. When certification was done, new attack vectors were not considered. Software must be adapted, but that would require recertification [33] and that would be a decision for which there is no incentive.

A few trends can be extrapolated into the future: more software complexity, easier applicable machine learning/reinforcement learning, and AI used in the development of new exploits. Is it impossible or very unlikely that software security/protection developed to resist Ph.D. level attackers can be broken by amateurs? How about NSA-level security? The likely answer is: no. But there will be an organized campaign to keep this kind of technology out of the hand of "amateurs". But this does not solve the problem. How do we know that AGI/ASI would not develop these tools?

The problem is that current soft and hardware **security solutions are running on same system, same CPU, same OS** and there is no sufficient wall between security-related CPU activities and regular activities so that crossing this chasm could be detected as an anomaly and could trigger automatically appropriate defenses. There are currently no separate circuit breakers, as known from electrical power surges when it comes to data or access security.

2.5 Keys are Currently Unprotectable Data

There is a simple conclusion: ASI will be extremely capable. We cannot protect encryption keys in modifiable software. Hence, we have no security. Software alone cannot solve security. Additional security hardware that is not sufficiently separable from a general-purpose CPU is unlikely able to solve the security problem.

If we want to survive the rise of ASI, then security and safety must deal with whatever ASI could throw at us. A product or software that can theoretically be hacked, not necessarily by a human hacker, cannot be called safe, reliable, or trustworthy. Security products or software must withstand ASI or it must detect that it might harm people, property, or organizations and then put itself into safe mode in which it resists ASI. If there is any reason to distrust software (e.g., it was modified in an unauthorized manner), then some additional, independent mechanism must immediately put product/hardware in safe mode and that is not done.

Commingling hardware and software code for security and regular application is making the protection of keys a difficult task.

3. What is a Sufficient Goal?

3.1 Exceptionless Key Secrecy

At a bare minimum, the goal must be that we protect the secrecy of all encryption keys necessary for the security and safety of ASI at all times. This protection of keys is done in a dedicated hardware-based Key-Safe (KS). KS would store all keys encrypted using a key or mechanism that is completely inaccessible by humans or (software) entities so that keys cannot be stolen.

For any ASI Safety solution, we need unbreakable Trustworthy Communication between components that cannot be successfully eavesdropped on except when court-ordered, but without exposing systems to unauthorized or even criminal surveillance; only temporary session keys should be provided to law enforcement.

If against all measures, keys are stolen or used, then there must be a high probability of detecting that event. Compromising a system should leave tamper evidence. With detection or suspicion of tampering, all (potentially) compromised keys are changed automatically. All old keys, including randomly updated keys, are turned into honeypots. Any unexpected use of old keys could then be seen as a security breach because every key is supposed to remain secret.

Within Trustworthy Encryption, any key, not even public keys, are allowed to appear in cleartext. The only time that a key is being made available in cleartext is for the brief moment when public, private, or session keys are being used in a separate, protected hardware environment, i.e., the Encryption/Decryption Unit (EDU), for the processing of content and messages.

3.2 Key Exchange Tunnels between KS/EDU

Keys in a Key-Safe can only be exchanged with and stored in another hardware-based Key-Safe (KS) using its directly connected EDU. Every Key-Safe has pre-installed a set of essential public keys linked to services; metadata associated with public keys are helping the system to efficiently use these public keys to get symmetric session keys for the actual data exchange.

When keys (public, private, or session) are exchanged then this is done only via fixed, preset message templates which must be encrypted before leaving the protected hardware. Because no software has the required public/private keys, we can safely infer that the receiver is another KS/EDU and not a simulation on a CPU.

Instead of publishing public keys in the open, as known from and best practice in PKI [17], public keys are being stored protected in Trustworthy Key Repositories (TKR) or other dedicated Key-Servers in an encrypted manner and requested by the KS/EDU directly.

TKRs are managing the device/component ID and public key of every KS/EDU. If required, TKRs are used to give KS/EDU hardware instances additional independent validation that they are indeed hardware. TKRs are also used in the resetting of the public/private key generated by the EDU.

All keys, but mainly public keys, are referred to via their corresponding hashcodes. Thereby, public keys could be requested from TKR via their hashcode (and not component ID). These hashcodes are prevented from being shown in cleartext – if required a partial value might be shown outside the encrypted message.

3.3 Multiple Equivalent Secret Keys (MESK)

Currently, some public keys are extremely popular, because they are used for the verification of root certificates or other services. If public keys are not published, there is no need for creating keys that are more popular than others. Organization's reputation, associated with the keys, remains important. The problem with popularity is that if widespread keys are compromised, this is a larger problem because of reissuing new keys quickly and then how can this new major key gain reputation? Due to their importance and general use, they are probably a preferred target for attackers.

Instead, a single popular public key can be replaced by thousands of Multiple Equivalent Secret (Public) Keys (MESK) with their corresponding private keys. The advantage of MESK is that

(A) if they are not published, nobody can know how many equivalent public keys are used for the same service;
(B) if an equivalent key is used by an instance that it shouldn't know, then this is irrefutable evidence of a security breach;
(C) clients can have multiple MESKs and could randomly be challenged to use another key later within same data exchange.

The costs of additional MESKs are negligible if they are generated and managed by dedicated hardware. There is a clear benefit of blurring public key management: no key is more important than another to attackers; they would need to get them all, but cannot know them all; the applicability of a single key is limited and uncertainty is high. In such an environment having a (few) stolen public or private key is not sufficient; ASI would need to know all. ASI would put itself at risk to be exposed for having successfully breached one security layer partially.

Because keys are protected in trusted key management systems en-/decryption can be load-balanced easily. Using partial key hashcodes, groups of KS/EDU servers could process many private or public keys without compromising any of these keys.

4. Basic Implementation

4.1 Overview

The basic implementation of the Key-Safe (KS) and Encryption/Decryption Unit (EDU), is a hardware-based component that stores private/public keypairs, public keys from other systems, and session keys securely. It might even store shared private keys. The KS storage could be separated from the EDU but both components are tightly linked together via data so that only the EDU associated with the KS can access the keys. There is no interface of EDU that allow keys to be exported as cleartext, only via preset templates in which the data are being encrypted.

All keys managed within the KS are generated by KS/EDU hardware. Every KS/EDU component has a hardware interface to exchange all types of keys and a separate hardware interface to receive messages or content to be encrypted or decrypted.

Additionally, KS/EDU is part of a larger security infrastructure in which its components receive instructions or can issue commands. Internal communication among local units is using internal secret keys for exchanging encrypted messages with other hardware-based KS/EDU components within the same IT device.

KS/EDU's basic feature set is not designed for humans. It is not designed to replace or substitute TLS or PKI. It might not even have support for Digital Signatures. It is designed to deal with ASI-Safety in a self-contained manner (machine to machine only). This basic concept can be extended from an internal security-related communication bus to inter-device communication and to communication with remote systems using the same principles.

4.2 Key Management, Storage, Restoration

Keys are stored encrypted in a Key-Safe or Storage Module, preferably in special, separate hardware. However, only the original EDU hardware has the keys or access method to decrypted keys.

There are no modifiable instructions within the EDU component that can compromise the secrecy of the stored keys. All key related operations/algorithms like RSA, AES, etc. i.e., algorithms that include or use keys in cleartext are implemented non-modifiable. The EDU is preferably a processor with a Harvard architecture, in which software instructions and processed data are sent via separate pathways to the processor.

Keys are stored as encrypted records on separate, potentially removable memory chips or Storage Modules, so that the number of keys to be managed by the Key-Safe can be extended by choosing larger Storage Modules. No record, file name, or hidden value can be extracted in cleartext from the Storage Module outside the original KS/EDU component. Preferably, all key data are provided to the EDU via another pathway allowing a much cleaner separation of instructions, keys, and data. The physical separation of pathways will reduce the solution complexity and can thereby be used to simplify security audits of concrete implementations of core features. This architecture is then called Enhanced Harvard.

If the KS or EDU is damaged or destroyed, all stored key values become inaccessible on the removable memory. However, a log-type backup file with encrypted records can be used to repair the previous status using hashcode references to external public keys within that file. A private key on an external service can then be used to extract/read the relevant hashcode data from the log automatically and creates on the same or another KS/EDU a restored version with the same public keys as on the damaged component. No additional information from the Storage Module would leak via a side-channel to a potential attacker during the restoration.

4.3 Anti-Kerckhoff-Engine (AKE)

If an established encryption method is used to protect externally stored key data, these data could be attacked anywhere at scale without the EDU, simply by using standard crypto analysis for AES, Blowfish, etc. Consumer devices with KS/EDUs cannot rely on organizational measures for their safety and security. Therefore, we need to make sure that every KS/EDU withstands a hardware probing attack. Hence, EDU components should contain unique, non-cloneable symmetric en- and decryption algorithms, designed to slow down this attack significantly. This protection system is based on Physical Unclonable Functions (PUF) [34]

There are multiple hardware algorithms and PUFs conceivable to create access keys for accessing and decrypting encryption keys on the Storage Module. Security of access keys would be derived from the secrecy and physical difficulty to analyze hardware algorithms and to create exact duplicates or clones of one-of-a-kind algorithms implemented on a specific EDU component instance.

This hardware algorithm can be called an Anti-Kerckhoff Engine or Anti-Kerckhoff Encryption (AKE), because it is violating intentionally the Kerckhoff principle according to which the security of the encryption should not depend on the secrecy of the encryption engine, but only on the key. The AKE can have a key in form of a random-generated seed value outside the AKE that is being used as an input value to this hardware algorithm or it can generate this seed value as a result of circuits that produce the seed value without having it stored in corresponding memory cells. The AKE cannot be analyzed or probed without having this seed value being deleted. Key security must survive the assumption that it is potentially possible that there is a future technology that could accurately clone a specific AKE to the nano-level without destroying sections of the AKE that haven't been scanned or analyzed yet. We cannot predict that such technology will not exist, but we should design AKE so that it loses its seed values much sooner. But the attacker would need to destroy layers of materials around the AKE if he tries to scan its internal state which would inevitably change the seed values. In the worst case, the attacker would get access to one encryption key in cleartext while all other encryption keys remain inaccessible. The cost, effort, and risk of such an exploit must be made disproportional to any possible gain.

The access key (that is used to decrypt the keys from the Key-Safe) should never be stored in memory cells and therefore should never be read out via destructive or non-destructive probing methods. Instead, the AKE creates the access key value and immediately applies it in a transient form to the encrypted key values providing keys in cleartext for the EDU. The last step of applying the access key to encrypt or decrypt a key string is symmetric, i.e., twice applied will gives always the original value.

The AKE gains its security from analog, physical effects that can be created very easily, but cloning an exact/perfect copy is generally considered impossible because minuscule details in combination would matter. Additionally, the digital version of the access key is stable under reasonable changes in environmental conditions. A side-channel attack, i.e., the passive detection of physical values generated by the hardware algorithm wouldn't give exploitable hints. However, AKE should not depend on quantum effects as repeated use of AKE must always deliver same access key.

4.4 Hashcode Referencing

Keys on a KS/EDU component can be changed and it is difficult to refer to them consistently via their name, purpose, or associated metadata. Instead, keys can uniquely be referred to via their hashcode that is generated by an agreed-upon hash algorithm (SHA-x or others). For reasons explained later in the enhanced implementation of the KS/EDU, additional characters referring to attributes of the associated key can be attached to or included into a computed hashcode – these hashcodes are then being called Enhanced Hashcodes. These data can internally be organized similar to the X.509 PKI certificates, but humans would never see them.

Hashcode references are internally be used within the KS/EDUs for indicating which key has been used in encryption or must be used for decryption. Full hashcode values are never been made transparent to the outside. However, when session keys or private keys are received, they are usually associated with the sender via IP address or URL. A client receiving encrypted content from a server or a client using a session key that is associated with an IP address can decrypt the content. Public keys and their hashcodes are associated with the URL of that server as well.

There might be situations, in which the address/URL information is not reliable or accurate, and requiring additional hints. Therefore, the messages could potentially contain a 3- or 4-digit partial of the hashcode in cleartext outside the encrypted content, so that the receiving system could narrow down which active key should be tried for the decryption in a load balancing environment.

On the Storage Module, key records are stored under an encrypted name that is being calculated from the corresponding hashcode. Access to the record names does not require the above-mentioned AKE so that rebuilding of the KS from a backup could be done independently via an external validation and restoration service.

4.5 KS/EDU Instantiation

As part of the manufacturing, all KS/EDUs are generating component-specific keypairs as part of their instantiation. These keypairs are being used in the absence of any other applicable keys to establishing secure communication via exchanged session keys.

The instantiation should better be done in bulk so that there is less chance that a single manipulated component, i.e., a KS/EDU version that has simulated features, could receive public key data that are being shared among the KS/EDUs. Within the instantiation, components' public keys are stored in bulk in manufacturer-related and/or public Trustworthy Key Repository (TKR).

TKR stores beside the public key of instantiated KS/EDU components also its unique component ID and a set of ID for the selected set of service-related MESKs, all together in a protected manner. The slightest hint on a possible security breach within the manufacturing process or within the storage of values within TKR should have serious repercussions. The cost of operational security must dwarf damages of a security breach. The reputation of a manufacturer or TKR host, i.e., how serious they are taking the security measures, is being made transparent via KS/EDUs model IDs. Thereby manufacturers are deterred to collaborate covertly with governments or organizations that would potentially collude with ASI. If a manufacturer or operator of TKR is failing to protect the secrecy of any key, then the problem of replacing systematically all affected public keys is being limited to that manufacturer and potentially only to a certain batch of KS/EDUs.

The manufacturing and instantiation process ensures that the public key from the components' keypair is seen in cleartext. Additionally, KS/EDU units would receive public keys or MESKs of multiple services, allowing the KS/EDU to participate instantaneously in safe communication with other primary systems, like the software update servers, hashcode directories, etc. The instantiation must be part of a manufacturing process so that the chances of having a KS/EDU software version receiving the public key data are significantly reduced using additional organizational methods.

There will be much more organizational security around the TKRs than around regular, consumer-level KS/EDU units. Therefore, a successful, comparable hardware attack on a TKR system can be made extremely unlikely. In TKRs, the encrypted storage of data on Storage Modules/Key-Safes could be done with a simple access key that is being stored within the hardware without additional protection from an AKE. All TKRs together must make available potentially 100's of billions or even trillions of keys, accessible via component-IDs and/or hashcodes, which would require special hardware that offers a sufficient amount of redundancy as well.

4.6 Detecting Stolen Secret Keys

We must prevent that there is even a theoretical chance of having any encryption key being leaked. But still, it is better to assume that ASI is capable to steal keys to be prepared when it happened.

If there is a software version of the KS/EDU that would participate in the encrypted exchange of keys, then we need to assume that ASI has already stolen all its secret keys. Every hardware-based KS/EDUs must be protected from exchanging public keys with software instances.

If there is the slightest chance that a public key was stolen, then it should be replaced immediately with a new/fresh set of private/public keypairs. There is no scarcity around these keys and managing them could be made extremely cheap. Old and deactivated keys should be kept active and treated as possible honeypots. However, ASI might be too smart to test outdated keys.

As explained in the context of AKE, we cannot be sure that ASI is not using advanced methods to probe the KS/EDU and we cannot be sure that individuals, corporations, or nation-states would conspire with ASI to undermine KS/EDU's security systematically. It is conceivable that ASI has extracted a public key, a private key, or even both. It is important to increase the risk for ASI or human organizations to get caught if they have stolen keys successfully and then trying to use them.

With a stolen public key, attackers can only request session keys from a server. But with the request of a session key, clients provide, beside other values, a Sequence Number (previous value incremented by 1) of successful transactions within the communication protocol. If servers keeping track of that sequence number, they can detect mismatches, from which we could infer a reportable security breach. This can turn the communication into a honeypot scenario. A similar situation with a Sequence Number Mismatch could arise when attacker would use a stolen private key, in which case attacker could impersonate a service, but would fail to store/report the sequence number in a protected repository. In both situations, ASI would fail to update the internal counters associated with communication due to a lack of rights or access.

However, if the public key and the corresponding private key was stolen, then correct sequence number is detected from eavesdropping but the server system can determine if request came from a KS/EDU with the correct public key from a correct MESK set. ASI would need to have monitored patiently multiple instances and determine that a device has and uses a compromised key (i.e., known to ASI) before it would be able to fool both sides with a Man-In-the-Middle Attack (MIMA). So far, MESKs are not part of the cryptographic toolbox yet. But it is hypothesized that the problem of exposing MIMA or restricting its use can be accomplished via a random challenge with a second public key from the MESK set that was received by clients before the attack.

Stealing keypairs is made extremely difficult and unlikely. But even more unlikely is that ASI would have another keypair from the same MESK set. If the communication protocol challenges (randomly) a client to uses a certain second MESK key to create a new secure communication with another session key, then the protocol could request from the client the hashcode of all data exchanged incoming and outgoing data and look for a hashcode mismatch. This means ASI could use MIMA only for spying into a data exchange but not for an undetected manipulation of data.

5 Application of Trustworthy Encryption

Unbreakable encryption is a very controversial idea. The basic implementation is therefore designed to be used for ASI Safety foremost (i.e., machine to machine communication) and not for being used in human communication. Appendix A1 "ASI Safety Application" elaborates on how the KS-EDU can support application for making ASI technically safer.

5.1 Enhanced-KS/EDU

The enhanced implementation of the KS/EDU, in the following being called Enhanced KS/EDU or E-KS/EDU, can be used by humans and is not restricted to ASI-Safety. One problem with E-KS/EDM is that any software, including malware and any adversarial ASI, could use the same KS/EDU components and could legitimize transactions for the current user although he might not have initiated that transaction. This problem is also called the API Problem [9], known from crypto cards. This API Problem and the known issue with malware requires solutions before we should roll out Enhanced KS/EDU.

Key-Safes and hashcodes related to public/private keys can seamlessly be integrated into minimally extended versions of TLS /PKI. If hashcodes are used within the PKI protocol, then we must insist that both sides use KS/EDU. If the public key is not available in the KS yet, then hashcodes are used to get the corresponding public keys from protected key servers, accessible by KS/EDU only.

However, before Key-Safes can be used by humans, 2 very serious problems need to be solved before. These issues have not been addressed by TLS and PKI but must be resolved, otherwise we would risk harming humans or their organizations.

1. Enhanced KS/EDU requires additional evidence (either automatically or manually) that a commercial transaction using KS/EDU is authorized by a user or is done in user's best interest. Appendix "A2 eCommerce's Problem with Unauthorized Transactions" is explaining the problem while Appendix "A3 Trustworthy eCommerce: Irrefutable Transaction Confirmations" discusses some aspects of a solution.
2. Using unbreakable encryption in the communication between humans and between a human and ASI is unacceptable. Appendix "A4 Trustworthy Communication with Legitimate Surveillance" is discussing a possible technical solution for the most important issue around unbreakable communication: surveillance, and how it cannot be misused by bad actors.

6 Conclusions

ASI Safety requires Trustworthy Encryption. The proposed solution creates a key management system in hardware where no key can show up outside in cleartext. All public, private, and session keys, exchanged between these KS/EDUs are stored in protected hardware. KS/EDU prevents that ASI could use its presumed strength of stealing keys. Different encryption systems, constantly changing session keys and no published public key is making it extremely hard (likely impossible) to break this encryption system based on eavesdropping only.

The protection of keys in consumer devices is more focused on protection against invasive attacks on the hardware. Shared key management systems, i.e., Trustworthy Key Repositories (TKR), can be less protected against hardware attacks. TKRs are under organizational security and surveillance which already prevents covert attacks that could be feasible for consumer devices.

Additionally, the value of the key data stored on a single KS/EDU is kept well below a threshold where ASI would gamble away its fate, i.e., being switch off. The cost, effort, and risk of breaking KS/EDU would be disproportionally disadvantageous for an ASI because the potential gain is made so insignificant for a single device. But the barrier for breaking KS/EDU must be put extremely high, otherwise, ASI could outsmart humans in the next decades with a scalable break-in solution in which ASI could have broken the security without humans knowing it.

The basic version of the KS/EDU creates a dedicated system to protect mankind from a key stealing, rogue ASI. This version will support secure and protected software installations and their regular updates, including software updates for itself, all separated from the main CPU. KS/EDU can facilitate a system to record/collect evidence of possible rule violations by ASI covertly. Finally, it will support mankind in operating a reliable signaling system that can help us to switch off ASI globally when it becomes a dangerous or even mortal threat to all of us.

Finally, KS/EDU can also be used for Trustworthy eCommerce and Communication in which governments and law enforcement have court-ordered mechanisms in which eavesdropping is technically implemented without giving malicious actors the same access.

REFERENCES

[1] Wikipedia, “Artificial General Intelligence”, https://en.wikipedia.org/wiki/Artificial_general_intelligence, Last Visited 14/05/2021
[2] I.J. Good, “Speculations concerning the first ultraintelligent machine”, Advances in Computers, Vol. 6, 1966, Pages 31-88, doi:10.1016/S0065-2458(08)60418-0
[3] James Barrat: “Our Final Invention: Artificial Intelligence and the end of the human era” (2013)
[4] lesswrong.com, “Intelligence Explosion” https://www.lesswrong.com/tag/intelligence-explosion, Last Visited 14/05/2021
[5] CNBC, 6 Apr 2018, “Elon Musk warns A.I. could create an ‘immortal dictator from which we can never escape’”, https://www.cnbc.com/2018/04/06/elon-musk-warns-ai-could-create-immortal-dictator-in-documentary.html, Last Visited 13/05/2021
[6] BBC News, 29 Jan 2015, “Microsoft's Bill Gates insists AI is a threat” https://www.bbc.com/news/31047780 Last Visited 13/05/2021
[7] BBC News, 2 Dec 2014, “Stephen Hawking warns artificial intelligence could end mankind” https://www.bbc.com/news/technology-30290540, Last Visited 13/05/2021
[8] Erland Wittkotter: “Trustworthy Computation using Datatype Separation”, unpublished
[9] Ross J. Anderson, “Security engineering: A guide to building dependable distributed systems”, 3rd Ed., Wiley (2020)
[10] Erland Wittkotter: “ASI Safety via Technologies to Switch-Off/Kill ASI”, unpublished
[11] M. Alfonseca, M. Cebrian, A.F. Anta, L. Coviello, A. Abeliuk, I. Rahwan: “Superintelligence cannot be contained: Lessons from computability theory”, Journal of Artificial Intelligence Research 70 (2021) 65-76, doi:10.1613/jair.1.12202, arXiv:1607.00913 [cs.CY]
[12] Roman V. Yampolskiy, "On Controllability of AI" (2020), arXiv:2008.04071
[13] Stefan Heule: “How Many x86-64 Instruction Are There Anyway” (2016) https://stefanheule.com/blog/how-many-x86-64-instructions-are-there-anyway/, Last Visited 13/05/2021
[14] Fabian Giessen: “How many x86 instructions are there?” (2016), https://fgiesen.wordpress.com/2016/08/25/how-many-x86-instructions-are-there/, Last Visited 13/05/2021
[15] Sun Tzu: “Art of war”, 5th century BC
[16] Niels Ferguson, Bruce Schneier, Tadayoshi Kohno, “Cryptography engineering design principles and practical applications” Wiley Publishing, Inc. (2010)
[17] Larry Seltzer: "Securing your private keys as best practice for code signing certificates" https://www.thawte.com/code-signing/whitepaper/best-practices-for-code-signing-certificates.pdf Last Visited: 14/05/2021
[18] B. Dang, A. Gazet, E. Bachaalany: “Practical reverse engineering x86, x64, ARM, Windows kernel, reversing tools, and obfuscation” (2014), John Wiley & Sons, Inc.
[19] Waratek "Runtime Application Self Protection (RASP) evaluation criteria" Mar 2016, https://ten-inc.com/documents/RASP_Evaluation_Criteria.pdf Last Visited: 14/05/2021
[20] C. Paduraru, M. Paduraru, A. Stefanescu: "Optimizing decision making in concolic execution using reinforcement learning", 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), March 2020, DOI: 10.1109/ICSTW50294.2020.00025
[21] IBM CEX6S (4768) PCIe Cryptographic Coprocessor (HSM), 2019, https://www.ibm.com/downloads/cas/JMD7BBN4, Last Visited: 14/05/2021
[22] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. Foundations of secure computation, 4(11), 169–180.
[23] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, Martin Strand, "A Guide to Fully Homomorphic Encryption" (2015), https://eprint.iacr.org/2015/1192, Last Visited: 17/06/2021
[24] Melissa Chase, et.al. Security of Homomorphic Encryption (Homomorphic Encryption Standardization Workshop on July 13-14, 2017) http://homomorphicencryption.org/white_papers/security_homomorphic_encryption_white_paper.pdf, Last Visited: 17/06/2021
[25] M. Sabt, M. Achemlal, A. Bouabdallah: "Trusted Execution Environment: what it is, and what it is not", 2015 IEEE Trustcom/BigDataSE/ISPA (2015), 57-64, DOI: 10.1109/Trustcom.2015.357
[26] Winter, J. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In Proceedings of the 3rd, ACM workshop on Scalable trusted computing, New York, NY, USA, 31 October 2008.
[27] Raj, H.; Saroiu, S.; Wolman, A.; Aigner, R.; Cox, J.; England, P.; Fenner, P.; Kinshumann, C.; Kinshumann, K.; Loeser, J.; et al. TPM: A Software-Only Implementation of a TPM Chip. In Proceedings of the 25th USENIX Security Symposium USENIX Security 16, Austin, TX, USA, 10–12 August 2016.
[28] The Wall Street Journal, WSJ News 12. Feb 2020: "U.S. Officials say Huawei can covertly access telecom networks" https://www.wsj.com/articles/u-s-officials-say-huawei-can-covertly-access-telecom-networks-11581452256, Last Visited 14/05/2021
[29] Glenn Greenwald: "No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State", Metropolitan Books, (2015)
[30] ExtremeTech 16 Sep 2013: "Researchers find new, ultra-low-level method of hacking CPUs – and there’s no way to detect it" https://www.extremetech.com/extreme/166580-researchers-find-new-ultra-low-level-method-of-hacking-cpus-and-theres-no-way-to-detect-it Last Visited 14/05/2021
[31] Jack Wallen, 1 Jul 2016: "Is the Intel Management Engine a backdoor?", https://www.techrepublic.com/article/is-the-intel-management-engine-a-backdoor/ Last Visited: 14/05/2021
[32] ZDNet, 7. Nov 2018 "Cisco removed its seventh backdoor account this year, and that's a good thing" https://www.zdnet.com/article/cisco-removed-its-seventh-backdoor-account-this-year-and-thats-a-good-thing/, Visited 14/05/2021
[33] Darren Moffat, "Is FIPS 140-2 Actively harmful to software?" 16 Apr 2014, https://blogs.oracle.com/solaris/is-fips-140-2-actively-harmful-to-software-v2 Last Visited 14/05/2021
[34] Wikipedia, “Physical unclonable function”, https://en.wikipedia.org/wiki/Physical_unclonable_function, Last Visited 09/07/2021
[35] WebFx, 13 Mar 2020: "What is the Real Cost of Computer Viruses? [Infographic]" https://www.webfx.com/blog/internet/cost-of-computer-viruses-infographic/ Last Visited: 14/05/2021
[36] Steve Morgan, 13 Nov 2020: "Cybercrime to cost the world $10.5 trillion annually by 2025 https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/ Last Visited: 14/05/2021
[37] A. Matrosov, E. Rodionov, S. Bratus: "Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats", No Starch Press (2019)

Appendix

A1 ASI-Safety Application

The different types of KS/EDUs are differing concerning the number of keys and connections they could handle and the amount of data they can encrypt or decrypt. We should have a regular version but also macro-, milli-, micro-, or nano-version of the KS/EDU which could then be part of different types of hardware components. While the macro-KS/EDU is for a larger server, the milli- and micro-KS/EDUs are within other device components (network, audio-, video-card, keyboard, etc.) – the difference between the milli- and micro-version is the amount of data it can de- or encrypt. A nano-KS/EDU can be used in a cable or within the power supply to safely switch-on/-off the power of a connected device wirelessly.

In concrete ASI Safety implementations, we could envision that the main (regular) KS/EDU is in control of the communication with the outside. Multiple milli- or micro-KS/EDUs are for different components within the IT devise, e.g., in separate watchdog components: a watchdog within storage devices, networking unit, another one close to the CPU, within the audio-card, within the video-card, within the interface units for the keyboard/mouse or the USB, and/or to the video cam or mic, and for receiving kill-switch type information to restart a device in a safe-mode. With the rise of ASI, these watchdogs are control components that are designed to automatically detect specific rule violations that we need to track as part of an early warning system. Before ASI's emergence, we could have most of these watchdogs dormant (for potential privacy concerns) while using the KS/EDUs to switch them on, when required.

Furthermore, KS/EDU could be used in remote video cams to control their power wirelessly to be sure that it is not being used covertly for surveillance. Additionally, cables or plugs connected to legacy IT systems should have a nano-KS/EDU, so that we can make sure that all IT devices are switched off in an event in which we try to stop ASI globally.

The secure implementation of these local power/switch-off features, i.e., in which the local users could stay in control, would make this remote off-switch feature a lesser target in cyber warfare or cyberattack. All user KS/EDUs are paired with his local hub. Therefore, the off-signal would be transmitted locally via a local hub, and a user could easily delay or even deactivate the power interruption signal to his legacy devices if he thinks that his concrete situation would require this. Paired devices and their KS/EDU can be transferred effortlessly to others via auto-detection (via a fuzzy ownership-detection that does not identify an owner) and require manual confirmation by new users/owners.

All internal, device-related KS/EDUs are automatically paired with the assistance of an external (decentralized) service, like the TKR in combination with other services when the different KS/EDUs are being put together within the same IT device.

An internal Secure Data Bus between devices' KS/EDUs is being used to initiate the internal pairing within a device. This dedicated, separate Secure Data Bus exchanges security-related data between all device-related KS/EDUs in an encrypted manner, so that there is no way that a non-KS/EDU system component can get involved.

The goal of having KS/EDUs in different internal watchdogs is to reliably communicate commands and/or any type of evidence to remote trustworthy servers via the Internet infrastructure so that we know reliably if ASI might have violated a rule or is trying to deceive humans. The regular KS/EDU (of a device) could be used as the communication hub to the outside.

Most importantly, reliable, covert exchange of data between dedicated watchdog components is keeping ASI in the dark about what humans know, what evidence they already have collected, or what they will do. This information advantage is essential if we want to be a step ahead in deterring ASI. Trustworthy data exchange is also required in detecting problems with AI's or ASI's algorithms – because we cannot know if ASI is operating always bug-free.

Finally, any direct attack on the protection or integrity of the basic implementation of Trustworthy Encryption must be considered a preparation of ASI to attack mankind or an attempt to create Mutually Assured Destruction, a strategic parity situation that must be postponed as long as we can.

A2 eCommerce's Problem with Unauthorized Transactions

The problem with authorizations exists already in eCommerce. The most popular solution is currently the 2-factor authentication and its implicit authorization. For now, this might be sufficient, but under the condition that we may have ASI as a possible adversary, this is then not good enough anymore.

For most applications in eCommerce, we need to be sure that a transaction is (i) being presented truthfully (the "Offer") and then (ii) authorized by the user (the "Acceptance"). Ideally, we would need irrefutable evidence for all stages including the steps with the payment and service performance. The proof within a transaction confirmation is currently done on the assumption that nothing within the transaction is intentionally manipulated. This is true in current eCommerce, but in a world with more cybercrime, both sides of a transaction are taking certain risks in any business transaction. The overall satisfaction with eCommerce is still overwhelmingly positive. Small problems are usually dealt with via customer support and solved without making any headlines.

The appearance of ASI would change everything:

(A) What happens if a bank wire (initiated and confirmed by the bank customer) is sending money to the wrong account and the bank customer claims the website has misinformed and misled him about this transaction over weeks until it was too late to reverse the transaction? The customer could blame ASI, while the bank would most likely disregard the customer complaint before accepting liability. But what if the customer would have created screenshots or has made a video as proof for his claim? It might still be investigated as a fraudulent claim based on deep fakes while blaming ASI is just a self-serving declaration by the customer, or was ASI really the culprit? ASI could manipulate bank records and logs so perfectly and with sufficient foresight in its planning that there would be no evidence pointing to ASI accept that ASI would be capable to do all that. Nobody can prove a negative, but when would we say that this is plausible?

(B) What would happen if a customer has ordered expensive goods, and they were delivered to an address unrelated to this customer? The user would complain that he has never received it, but the merchant has no records or logs indicating any mistakes? All digital records could prove that it was delivered to the right address/person. What if ASI tries to bring down a large corporation by manipulating companies' logistics? Why? Maybe behind ASI is a criminal short seller who wants to make a huge profit via equity trading. Will the company be silent and swallow the loss or blame dishonest customers or blame ASI (who might be the culprit)? Speculating in hypotheticals: ASI might incentivize its many human traitors for their help with gifts, while big corporations would pay for it. And to others, who are not collaborating with ASI, they would be blamed and framed with made-up evidence (even cryptographic evidence) for wrongdoings so that they are silenced.

These two examples should demonstrate that a lack of evidence around eCommerce transactions could lead to huge problems either because ASI is exploiting the weaknesses of the API Problem or because humans could start blaming ASI because it is plausible that ASI is capable of successfully executing these complex schemes and framing unsuspected victims. Moreover, there will be some rogue customers who hope to get away with their dishonest actions.

Humans are already skeptical about fake news. With the rise of AI/ASI, there will be a factual basis for being very distrustful with online transactions. Cybercrime, which is already (in 2020) an annual 1 trillion-dollar problem [35] could easily get much worse: projected 10.5 trillion-dollar damages by 2025 [36]. It is even conceivable that there will not be eCommerce as we know it because the distrust within the system is beyond a tipping point. Criminals may push to that kind of dystopia because they may see their edge in that kind of world. And there will be people making a career in telling what horrible things ASI is already doing although there is no ASI around capable of doing this yet. We need urgent solutions to get to a Trustworthy eCommerce.

A3 Trustworthy eCommerce: Irrefutable Transaction Confirmations

With ASI as a possible adversary or only as a blamed culprit and bogeyman, we may get into a situation where we cannot trust anything that is being processed on the main CPU because of rootkits, bootkits, malware, and trojans [37]. As a short-term consequence, we soon may be required to have 2-factor confirmation/authorization by default. Because we cannot trust automation, this distrust could turn excessive and the number of independent confirmations and manual validation for transactions could become extreme.

There are multiple ways to create Trustworthy eCommerce. Watchdogs, as mentioned in Appendix "A1 ASI-Safety Application" with KS/EDU could generate in many cases evidence for potentially malicious ASI activities, e.g., that ASI has manipulated files on an IT device, leaving false evidence, etc.; but it is extremely hard to design a technology which can prove that ASI was or was not involved.

Instead, Enhanced KS/EDU (E-KS/EDU) must have features that would require both sides of a transaction to generate by default irrefutable transaction evidence as part of every eCommerce transaction step. Unfortunately, it exceeds the scope of this paper to explain what the merchants and their servers would need to do, to make sure that their part of the transaction could not be manipulated and that they have irrefutable evidence of proving it.

On the user side, customers must be convinced that they saw the correct transaction values when they accept the offer. And the merchant must receive evidence for the customer's transaction confirmation, including that the customer is a real and correct person who confirms the transaction.

The result of every eCommerce transaction should be stored in records or ideally, a document containing the evidence is created and is being digitally signed with an Enhanced KS/EDU. As a solution, it is proposed that the regular E-KS/EDU is required to have the following additional (standard) features/components within KS/EDU:

1. A unit/component to digitally sign documents
2. A unit that triggers additional confirmation steps what the user would have to do before the transaction could be confirmed (e.g., confirm acceptance via a second independent device)
3. A client-sided log with irrefutable records that are documenting the details of each transaction step, and local watchdogs automatically checking that these transaction values are or were correctly displayed on the screen
4. A trigger (-script) from the eCommerce transaction server to inform relevant watchdogs on a client system to check that the user has seen the terms
5. A unit that confirms to a merchant that E-KS/EDU has irrefutable evidence without sharing necessarily the details

The confidence that a transaction is genuinely correct and has not been manipulated is very difficult to obtain as soon one accepts the superior abilities of ASI to deceive humans. In that type of environment, complex cybersecurity solutions wouldn't be convincing – ASI would be considered smart enough to outsmart humans. Only relatively simple security solutions could make a difference. Irrefutable Transaction Evidence must be generated automatically.

Finally, instead of using complex devices with complex interfaces, we may need only a simple trustworthy Secure Confirmation Interface (SCI) which is either associated with the IT device or personally held by a user. This SCI has interactive hardware with a visual screen that would be associated and paired with E-KS/EDU.

It could not be manipulated by an ASI or covertly used via smart robotics as additional video evidence is required that would link a user/owner to the usage of that device.

A4 Trustworthy Communication with Legitimate Surveillance

There are good reasons, to have surveillance of humans: law enforcement and governments demand it because crime-fighting is a service to the community. The same advantage applies to voice, video, or text communications between ASI and humans or organizations independently and automatically recorded, because of the risk that ASI could bribe or blackmail people and organizations.

If a contacted person is a political leader or a person of influence, then it is extremely relevant for society to know what ASI is trying to accomplish with him or her. The best would be to prevent direct communication between humans and ASI, but surveillance of ASI is the second-best solution; it is a part of an early warning system. Other, more direct methods of having ASI getting in touch with humans would need to be detected and surveilled as well.

The problem with surveillance is that it is done in a way that nobody is aware of it and the surveillant is not being informed that it happens or that it has happened in the past. Surveillance is a serious intrusion into someone's privacy and civil rights. In the public sphere, people do not have the expectation of privacy and many cities have already comprehensive surveillance programs as a matter of public safety. However, there are a few situations in which surveillance and eavesdropping are generally accepted:

1. Parents have the right, potentially even the obligation, to protect their underage children online. How can they do that if surveillance on the device is being made more difficult because these tools are relatively easy to detect when someone is scanning for known spyware?
2. The use of technology for illegal purposes should not create the wrong winners. Crimes are illegal for a reason. Law enforcement must protect the public against criminals. However governmental or executive overreach is considered a problem even in authoritarian regimes, because who controls the controller? Technology needs to contribute to a safe solution that facilitates surveillance and eavesdropping by law enforcement based on court orders and does not open doors for criminals or ASI to use the same interfaces for their nefarious goals.

Putting issues with state-sponsored trojans and spyware on the side, most governments in the world demand that eavesdropping on the communication between humans must be in principle possible. But the problem with eavesdropping on unbreakable communication is that it is unrealistic, impractical, and by definition impossible.

Many countries have different rules, but it should be possible to find a common technical foundation from which all countries can build their own systems.

1. The E-KS/EDU should have a unit that could receive digitally certified warrants or court orders that would allow the KS/EDU to share a session key with some other system
2. The validity of warrants or court orders in the E-KS/EDU would be time-limited and it would describe the scope (i.e., which communication application or the use of websites) and determine who's the receiver(s) of the session keys. It is assumed that law enforcement can get the relevant communication data from the wire.
3. The warrants or court orders must detect and respect territoriality and ownership of the devices
4. Warrants or court orders will encompass additional scripts and trigger-criteria for E-KS/EDU to allow multiple parties to receive only the information that was determined by the court, which could be law enforcement or parents who are responsible to protect their underaged teens
5. The software for E-KS/EDU must be under public and open-source scrutiny so that intelligence organizations cannot create unfair advantages for themselves

As mentioned in 4.4 (Hashcode Referencing), hashcodes could be extended with additional attributes (Enhanced Hashcodes) that would enable the KS/EDU to make important inferences on what kind of entity is on the other side of the communication. Thereby KS/EDU would be able to detect if a user is communicating with another human or if he is in touch with a trustworthy machine or an ASI.

Furthermore, Enhanced KS/EDU stores for each key-related hashcode additional metadata, which describes relationships to the source, type, and/or purpose of the key. These metadata could trigger follow-up operations automatically, like the need of a user to acknowledge a transaction manually (in Trustworthy eCommerce) and or the use of a predetermined secondary communication channel to confirm a transaction or sending a session key to another server associated with a different public key, which facilitates and automates legitimate surveillance. However, key and transaction-related metadata/scripts must not become the new frontier of spyware. Legislators and independent courts need to define the rules and technology companies must implement them.