Attacks and Defenses for JTAG

Kurt Rosenfeld and Ramesh Karri
Polytechnic Institute of NYU [1]

Abstract

This article addresses some security issues surrounding JTAG. We look at the threat of a malicious chip in a JTAG chain. We outline attack scenarios where trust in a digital system is downgraded by the presence of such a chip. To defend against this, we propose a protection scheme that hardens JTAG by making use of lightweight cryptographic primitives, namely stream ciphers and incremental message authentication codes. The scheme defines four levels of protection. For each of the attack scenarios, we determine which protection level is needed to prevent it. Finally, we discuss the practical aspects of implementing these security enhancements, such as area, test time and operational overheads.

1 Introduction

JTAG[1] is the dominant standard for in-circuit test. It has been in use for 20 years and, like many mature standards seen in the landscape of information systems, JTAG was conceived with a friendly environment in mind. It was designed to handle the natural enemies of digital systems: faults in design, fabrication, packaging, and PC boards. It has been quite successful, largely due to the economy and flexibility of the system, and has been extended in various directions. It has evolved into the de facto method for in-circuit configuration and debug. The companion standard IEEE 1532 [2] has extended it even further to support on-board programming. While JTAG has great utility in friendly environments, in even moderately hostile environments it can lead to undesirable forms of exposure.

This article discusses attacks and defenses for JTAG-enabled systems. Our goals are to provide a practical path toward better security while maintaining strict compliance with the IEEE 1149.1 JTAG standard, all without significantly affecting test economics. We do not regard JTAG as a security threat. Indeed, the presence of JTAG on a board enables stronger platform security than would typically be achievable in an equivalent system without JTAG. Our view is that security problems arise when there is a discrepancy between what people expect and what assurances the system actually provides.

JTAG is a synchronous serial link intended for system management tasks. As shown in Fig 1, it supports daisy-chaining of an arbitrary number of devices. A single master device controls the protocol state of all other devices on a chain. From a standpoint of functionality, the order of the devices in the chain does not matter. Shared wiring and freedom over the order of the devices reduce the burden on the PCB designer, which translates into lower cost for JTAG compared with other in-system test and management solutions. In section 3, we see that the shared wiring and ordering of devices both have security implications.

The concept of JTAG does not preclude providing protection and assurances, but it is typically deployed in a minimal form which provides little or no protection. We will describe several ways in which an attacker can exploit a typical JTAG deployment to achieve his goals. Then we will review the aspects of JTAG that are relevant to the execution of the attacks. Section 4 surveys some of the prior work in this area, both on the attack side and on the countermeasure side. Following that, we will present defenses with the objective of improving security without incurring heavy extra costs.

[1]This work was supported by NSF grant 0621856. We also would like to thank Linda Lieu for her valuable help with the project.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

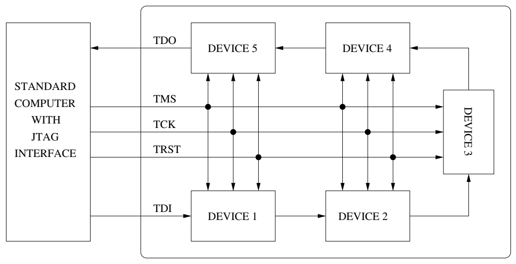

Figure 1: The typical deployment of JTAG is a chain of several devices on a printed circuit board. Each device may come from a different vendor. The test mode select (TMS), test clock (TCK), and test reset (TRST) signals are typically common to all chips. The test data in (TDI) signal and test data out (TDO) signals loop through the chips. The path returns to the source, which is usually a either a PC or an embedded microcontroller, functionally called a "system controller."

2 JTAG Overview

The JTAG protocol defines a bidirectional communication link with one master and an arbitrary number of slaves. The master can initiate an interaction with any of the slaves. Slaves never initiate interactions. A chip is JTAG-enabled by the presence of two essential physical components. The first is the test access port (TAP) controller, which is a state machine that interprets the JTAG serial protocol. The other physical component is the boundary scan register (BSR), which is a register that is interposed between the core logic of the chip and the I/O modules of the chip. The BSR can be arbitrarily wide, but can be preset and read serially. JTAG has three basic modes: BYPASS, EXTEST, and INTEST.

2.1 BYPASS Mode

In BYPASS mode, the serial bit stream is copied from the TDI pin to the TDO pin, delayed by one cycle of the test clock, TCK.

2.2 EXTEST Mode

In EXTEST (EXternal TEST) mode, the BSR is connected to the pins of chip. The BSR specifies the values that are to be asserted on the chip's output pins. The BSR captures the values that are present at the chip's input pins. The data in the BSR are subsequently shifted out through the TDO pin. The TDO signal is either routed back to the master directly or through other JTAG-enabled chips.

2.3 INTEST Mode

In INTEST (INternal TEST) mode, the BSR is serially loaded from the TDI pin and the contents of the BSR are applied to the terminals of the internal logic of the chip. The BSR captures the values that are present at the output terminals of the internal logic of the chip.

The JTAG standard requires supporting the basic instructions. Implementors are free to add their own instructions, and for these user-defined instructions they are free to define the semantics more or less arbitrarily. Whatever the instruction, the protocol follows the same basic pattern: Enter the SHIFT_IR state → Shift the instruction into the instruction register (IR) → Enter the SHIFT_DR state → Shift the relevant data into and/or out of the data register (DR).

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

3 JTAG Attacks

Before discussing JTAG-based attacks in general, we start by describing the reference platform we use for testing JTAG attacks and countermeasures. The platform was designed to be simple, yet representative of most of what one sees in typical JTAG implementations. For flexibility in our experiments, all of the JTAG TAPs in the system are *soft*, meaning that they are implemented as software on a microcontroller or as user-defined logic in an FPGA. This enables us to modify the TAPs, both to add attack logic and to add security enhancements.

The reference platform as shown in figure 2 performs the functions of an electronic combination safe. The benign use-case of the system is that a user approaches the safe, enters the combination, the door opens, the user places something in the safe, closes the door, and walks away. The circuitry involves three JTAG-enabled components.

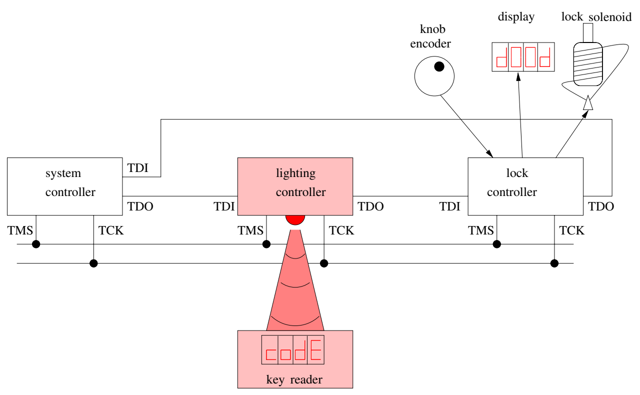

Figure 2: The reference platform demonstrates a sniffing attack and side-channel leakage of the sniffed secret. The system performs the functions of an electronic safe. The system controller initializes the key register in the lock chip at system startup time. This key is the correct combination for the safe. The lighting controller sniffs the key while passing it through and leaks it.

1. The **system controller** initializes volatile configuration registers in the lighting controller and the lock controller.
2. The **lighting controller** provides aesthetically pleasing patterns on the panel by controlling several multicolor light emitting diodes.
3. The **lock controller** performs physical access control. It takes in the combination from the user, compares it with the correct combination and unlocks the door of the safe if the combination is correct.

The lock controller is implemented as an FPGA. The system controller and lighting controller are implemented as microcontrollers. The lock controller has a key register which holds the correct combination. The system controller initializes this register at power-up, and also at run time if the combination is changed using a custom LOAD-LOCK-COMBINATION instruction in the lock controller. This LOAD-LOCK-COMBINATION operation on our reference platform is representative of many security-critical JTAG transfers that occur in real systems.

We augmented the JTAG TAP in the lighting controller with hostile functionality that tracks the state of the downstream TAP on the lock controller chip. When a LOAD-LOCK-COMBINATION

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

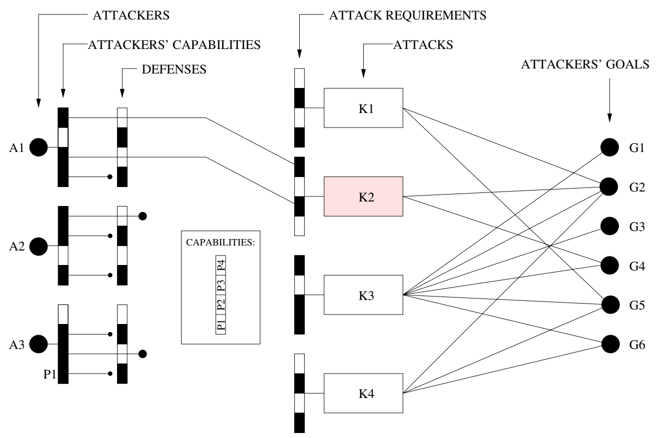

Figure 3: Conceptual security model: A set of attackers A1, A2, and A3 have a set of goals G1 through G6. Each attacker has a set of attack capabilities, some or all of which are masked by defenses that are in place. There is a set of attacks, K1 through K4, each of which requires a certain set of unmasked attack capabilities. Each attack can be used to reach some set of goals. This example shows that attacker A1 can achieve goals G2 and G4 since it has capabilities P2 and P4, which are the requirements for attack K2. Attackers A2 and A3 do not have sufficient unmasked capabilities to execute any of the attacks.

instruction is performed on the lock chip, the lighting chip obtains a local copy of the key that was sent, and leaks it by varying the pulse width modulation (PWM) dimming waveform. We implemented a reader that optically receives the key as it is leaked by the lighting chip and displays it on a set of seven-segment LEDs.

The attack scenario is that the hostile party has infiltrated the supply chain of the safe maker, supplying lighting chips with the aforementioned key-leaking functionality. The attacker then visits one of the safes after it has been deployed in the field and uses the key reader to obtain the combination (as set by the customer).

A more general model of the JTAG security landscape is shown in figure 3. The purpose of this model is to provide a way of analyzing the security risks associated with a JTAG deployment. A system can have multiple potential **attackers**. For example, one potential attacker of a system is the manufacturer of one of the chips. Another potential attacker is a hostile end user. Each attacker has a set of **capabilities**, some subset of P1 through P4. For example, an attack capability is to be able to sniff the data that is sent through the data lines. Some of the attackers' capabilities might be blocked by **defenses** that are in place. For example, the attacker might have access to sniff the bits on the JTAG bus. If those bits are encrypted, sniffing attacks will not work. There is a set of possible **attacks**, each of which has **attack requirements**. For example, an attack to capture the configuration stream of an FPGA as it is being programmed over JTAG requires the ability to sniff the JTAG data line. Each potential attacker will have a set of attack capabilities that are not masked by defenses that determines his set of possible attacks. Finally, we consider the **attackers' goals**. One possible goal an attacker might have is to cause a denial of service. Another possible goal might be to clone the system.

Returning to the example of our reference platform, we can see how the details of the scenario map onto the model. The attacker is the provider of the lighting chip. Because of the lighting chip's position in the JTAG chain, the attacker has the attack capability of sniffing the TDI data that goes to the lock chip. Since no encryption is used, the sniffing capability is not blocked. Because of the lighting chip's connection to LEDs, it has the capability of leaking data over an optical side-channel. These two unblocked capabilities enable the attack of capturing and leaking the data that passes from the system controller to lock chip. This attack serves the goal of obtaining unauthorized access to the contents of the safe.

Having looked at the specifics of one attack, we examine five basic JTAG-based attacks. The goal here is to see the general range of possibilities, not to exhaustively list all possible scenarios. After examining these attacks in terms of the security model in figure 3, we will be able to see what defense mechanisms we need.

In a given attack scenario, the attacker will possess a limited set of capabilities:

- sniff the TDI/TDO signals
- modify the TDI/TDO signals
- control the TMS and TCK signals
- access keys used by testers

The system topology in all of the examples is the same as shown in figure 1. We assume throughout that the attacker is capable of providing or otherwise controlling the attack chip. An attacker can combine one or more attacks to achieve his goals. Some attacks have already been carried out against real systems as discussed in section 4.

3.1 Sniff Secret Data

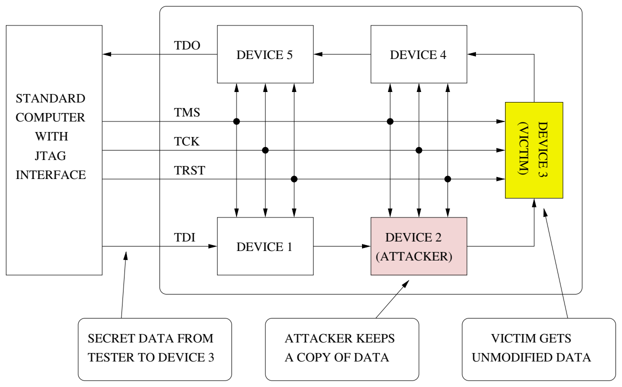
STANDARD COMPUTER WITH JTAG INTERFACE
TDO
TMS
TCK
TRST
TDI
DEVICE 5
DEVICE 4
DEVICE 3 (VICTIM)
DEVICE 1
DEVICE 2 (ATTACKER)
SECRET DATA FROM TESTER TO DEVICE 3
ATTACKER KEEPS A COPY OF DATA
VICTIM GETS UNMODIFIED DATA

Figure 4: The attacker obtains secret data by sniffing the JTAG data path.

The sniffing attack is shown in figure 4. The attacker's goal is to learn a secret that is being programmed into victim chip using JTAG. An additional requirement for this attack is that the victim chip is downstream from the attack chip on the same JTAG chain.

As shown in figure 4, attack chip exhibits a false BYPASS mode which is externally identical to true bypass mode, but which parses JTAG signals. When the secret is programmed into victim chip, the attack chip captures a copy. The secret is then delivered to the attacker either through JTAG interrogation of attack chip in the field, or through a side-channel. Alternatively, the attack chip might directly make use of the sniffed data.

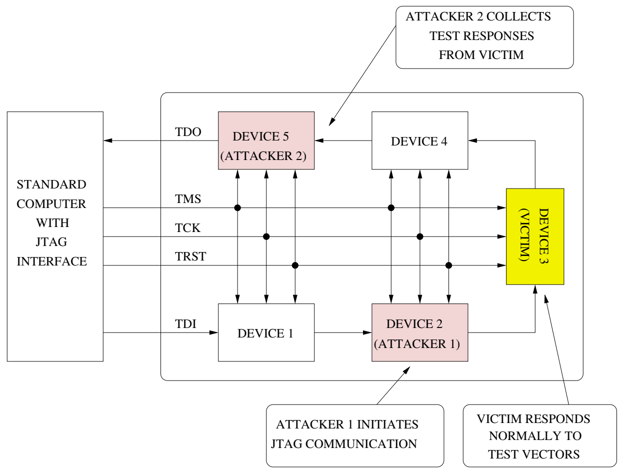
ATTACKER 2 COLLECTS TEST RESPONSES FROM VICTIM
STANDARD COMPUTER WITH JTAG INTERFACE
TDO
DEVICE 5 (ATTACKER 2)
DEVICE 4
TMS
TCK
TRST
DEVICE 3 (VICTIM)
TDI
DEVICE 1
DEVICE 2 (ATTACKER 1)
ATTACKER 1 INITIATES JTAG COMMUNICATION
VICTIM RESPONDS NORMALLY TO TEST VECTORS

Figure 5: The attacker obtains an embedded secret by forcing test vectors onto the JTAG lines.

3.2 Read Out Secret

The read-out attack is shown in figure 5. Here, the attacker's goal is to learn a secret that is contained in the victim chip. We assume that the attacker is capable of using I/O drivers in the upstream attack chip (attacker 1) to forcefully control the TMS and TCK lines. The details of forcing these signals are discussed in section 3.6. An additional requirement for this scenario is that the attack and victim chips are on the same JTAG chain with the victim chip sandwiched between the attack chips.

The upstream attack chip forcefully acts as JTAG bus master, and performs a scan operation on victim chip to access embedded secrets. Downstream attack chip collects the secret as it emerges from TDO of victim chip. It can be used by "embedded attackers" as described, or it can be used by an attacker who can attach external hardware to the system under attack.

3.3 Obtain Test Vectors and Responses

This attack is shown in figure 6. The attacker's goal is to learn the vectors that are used to test victim chip, and the normal responses. A requirement for this attack is that the attack chip is downstream from the victim on the same JTAG chain.

The attack chip waits for the victim chip to be tested by the authorized party. The attack chip collects the test vectors as they are shifted out of the instruction register, and collects the responses as they are shifted out of the selected register (BSR or others) on their way back to the tester. Knowledge of the test vectors and responses helps an attacker further infiltrate a system by providing trojaned parts that pass normal tests.

3.4 Modify State of Authentic Part

The attacker's goal is to modify the state of victim chip. We also assume that the attacker is capable of inserting strong I/O drivers in the attack chip to forcefully control the TMS and TCK lines. An additional requirement for this attack is that the victim chip is downstream from the attack chip on the same JTAG chain.

The attack chip takes control of the TMS and TCK lines, and puts the JTAG TAP of the victim chip into a state where it can shift data in, thereby setting the state of registers within victim chip,

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

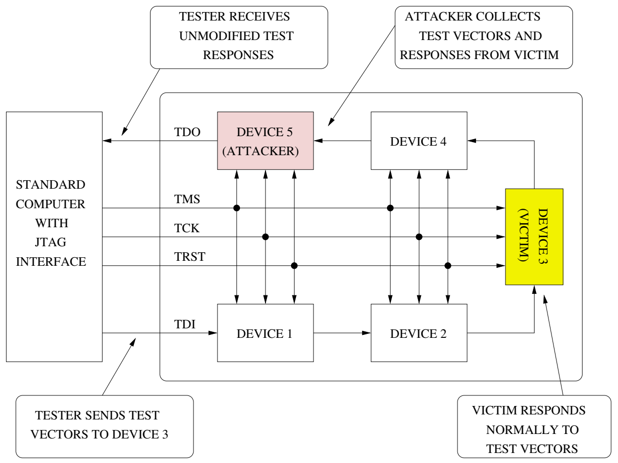
TESTER RECEIVES UNMODIFIED TEST RESPONSES
ATTACKER COLLECTS TEST VECTORS AND RESPONSES FROM VICTIM
STANDARD COMPUTER WITH JTAG INTERFACE
TDO
TMS
TCK
TRST
TDI
DEVICE 5 (ATTACKER)
DEVICE 4
DEVICE 3 (VICTIM)
DEVICE 1
DEVICE 2
TESTER SENDS TEST VECTORS TO DEVICE 3
VICTIM RESPONDS NORMALLY TO TEST VECTORS

Figure 6: The attacker obtains a copy of the test vectors and normal responses of a chip in the JTAG chain. This can be a passive attack.

including registers that affect the normal operation of victim chip.

3.5 Return False Responses to Test

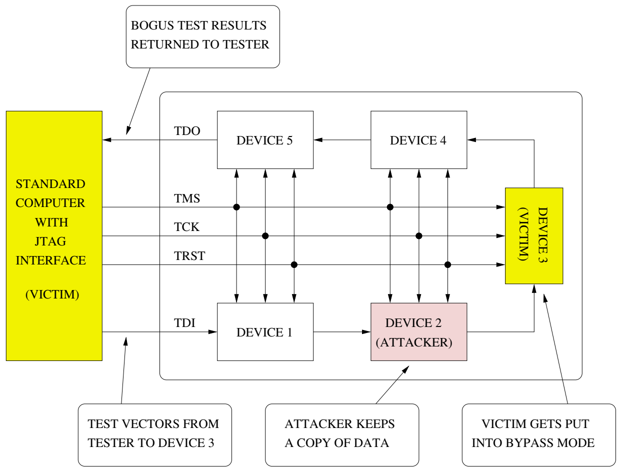
BOGUS TEST RESULTS RETURNED TO TESTER
STANDARD COMPUTER WITH JTAG INTERFACE (VICTIM)
TDO
TMS
TCK
TRST
TDI
DEVICE 5
DEVICE 4
DEVICE 3 (VICTIM)
DEVICE 1
DEVICE 2 (ATTACKER)
TEST VECTORS FROM TESTER TO DEVICE 3
ATTACKER KEEPS A COPY OF DATA
VICTIM GETS PUT INTO BYPASS MODE

Figure 7: The attacker can intercept test vectors that are sent to another chip, and can send false responses to the tester.

The false responses attack is shown in figure 7. The attacker's goal is to deceive the tester about the true state of victim chip. An additional requirement for this attack is that the victim chip is downstream from the attack chip on the same JTAG chain.

The tester attempts to apply test vectors to the victim chip, which is not the first chip in the JTAG chain. To do this, the tester attempts to place the other chips into BYPASS mode. The attack chip ignores this request and intercepts the test vectors, while instructing victim chip and other downstream chips to enter the BYPASS mode. Attack chip can then transmit the bogus test responses back to the tester.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

3.6 Forcing TMS and TCK

Whether an attacker can forcefully control TMS and TCK depends on various factors such as

- strength of output drivers on JTAG master
- strength of output drivers of attack chip
- presence of buffers in the TMS and TCK lines
- presence of series output resistor (typically 100 ohms)
- topology of JTAG bus (star or daisy chain)
- physical layout of JTAG bus
- the input logic threshold and hysteresis, if any

For the attacker to successfully hijack the TMS and TCK lines, he must be able change the voltage seen by the victim's TMS and TCK input pins. This change must be sufficient for the voltage to cross the logic threshold. If the JTAG is set up in a star topology, with separate TMS and TCK lines for each chip, this attack will impossible. However, it is quite common, in practice, for system designers to take advantage of the economy of wiring JTAG in a daisy chain topology. Designers can put buffers at various points in the JTAG wiring. There are various reasons for doing this, including noise immunity and fanout. However, these buffers add cost and complexity, and are not required by the JTAG standard. It is common practice to add a resistor, typically 100 ohms, in series with each of the outputs of the JTAG master. This resistor reduces ringing on the line by providing series termination and/or slowing the slew rate. In systems where this series resistor is present there is a reduction in the amount of current that must flow in order for the hijacker to force a TMS bit, or to force a TCK edge. The strength of the drivers in the master and in the attacker are also relevant. If the attack chip is an ASIC, the attacker can specify essentially any strength drivers. If the attack chip is an FPGA where the attacker controls the configuration bits, the attacker can program the I/O block to use a high-current I/O standard. If the attacker can control the PCB design, he can bond multiple pins of the attack chip together to form a mega driver. Finally, if the TMS and TCK lines are simply bussed around without buffers, multiple attackers can gang up to overpower the master. These last possibilities are included only for completeness. Next, we present practical experimental results testing the basis of the JTAG hijacking attack under normal conditions.

3.6.1 Experimental Validation

Our JTAG hijacking experimental setup is made of three chips: a system controller, a victim, and a hijacker. The system controller is a Philips 8052 microcontroller configured to bit-bang JTAG using port 1 directly, not through an I/O expansion chip (8255, etc.), nor through any resistor. The victim is a Xilinx Spartan 3e FPGA. The attacker is a Spartan 3 FPGA. The 8052 is programmed to keep TCK low all the time.

The experiment tests the hypothesis that the attacker can raise the TCK line sufficiently to edge-trigger the victim chip. We programmed the victim chip to count TCK transitions and to display the count on a set of seven-segment LEDs. This way, we were able to confirm the extent of the attacker's control over the TCK signal. We programmed the attack chip to send 1000 pulses. Comparing 1000 with the number of TCK transitions counted by the victim chip, we confirmed

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

that the Spartan 3 attack chip easily overpowers the 8052 system controller. In multiple runs, the victim always showed the correct count.

When one chip overpowers the output of another chip, there is a risk that the output driver circuitry of one or both chips will be damaged. In the experiment, we minimized the chance of overheating either chip by using a TCK waveform with a 80 nsec pulses and a 0.0015% duty cycle. We left the setup running for over an hour and saw no evidence of any sort of damage to any of the chips.

3.6.2 Common I/O Driver Characteristics

Electrically, DC models be used can represent the two output drivers fighting. Each driver will have an I-V curve for its logic-low state and an I-V curve for its logic-high state. The intersection of the logic-low I-V curve of the 8052 and the logic-high curve of the Spartan 3 gives the voltage at the node where their outputs meet. This is similar to solving for the quiescent point of an amplifier using load-line analysis. Using a pulse method, we obtained the I-V curves of the output drivers

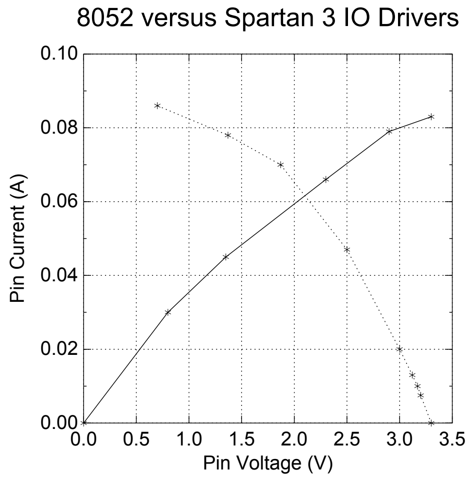

Figure 8: A Philips 8052 (the system controller) was programmed to keep one of its pins low. The I-V curve for this output driver was extracted using a pulsed I-V measurement. A Xilinx Spartan 3e was programmed to keep one of its pins high. This pin's I-V curve was also extracted. The result is shown. The solid line is the FPGA; the dashed line is the microcontroller. The intersection is at 2.1V, exceeding V_{IH} for most 3.3V logic.

of our 8052 and our Spartan 3. From these curves, we solved for the expected voltage when the drivers fight. The critical issue here is whether this voltage exceeds V_{IH} of the victim chip. In our experimental platform the receiving chip is a Spartan 3e using the 3.3 volt CMOS IO standard where V_{IH} is 1.1 volts and there is no input hysteresis. The Spartan 3 easily overpowers the 8052 and can force the TMS and TCK signals. In our measurements, the pulse width was 80nsec, with a repetition period of 1.3msec. This low duty cycle, 0.0061%, allows hostile JTAG communication but results in a worst-case dissipation of only $22\mu W$ in the output driver of either chip, not enough to cause damage.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

3.6.3 PCB Layout Effects

We discussed the TMS and TCK hijacking in terms of DC characteristics such as I-V curves and logic thresholds. We now examine a transient phenomenon that allows even weak hijackers to control the bus regardless of the strength of the master.

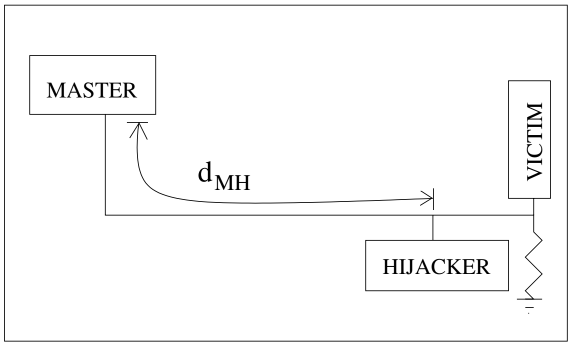

Figure 9: A length of PCB wiring connects the hijacker to the JTAG master. This allows the attacker to inject short pulses onto the wiring without being hindered by the master.

As seen in figure 9, there is typically a PCB trace of some length connecting the master's JTAG signals to each of the slaves. The trace length from the master to the hijacker, d_{MH}, results in a round-trip wiring delay of at least $rtt = \frac{2d_{MH}}{v_{prop}}$, where v_{prop} is the propagation speed of a pulse in the PCB trace. This propagation speed is typically around $2 * 10^8 m/s$. If we assume a PCB trace of 30cm, we obtain a round-trip time of at least 3ns. Even if the master can present a zero-ohm impedance at its end of the transmission line, each time the hijacker applies a $0 \rightarrow 1$ step to the TMS or TCK line, a pulse at least 3ns long will appear at the victim. This is sufficient to control a JTAG tap. We experimentally verified this phenomenon using the setup shown in figure 9, wired together using short sections of CAT 5 cable, $Z_o = 100\Omega$. The hijacker can control the victim's TAP even when the TMS and TCK lines are shorted to ground at the master's end.

4 Prior Work on Attacks and Defenses

Yang, et al.[3] showed that JTAG boundary scans enable an attacker to read secret data out of a chip. They use a hardware DES implementation as their example. Satellite television receivers make use of trusted hardware to prevent non-paying users from viewing the encrypted broadcasted content. There is an active underground effort[4][5] to hack satellite TV boxes by extracting their keys using JTAG[6]. This has been quite successful and continues.

On one hand, it is tempting to suggest that JTAG is a security risk and should be excluded from trusted hardware, but untestable hardware is a prohibitive business risk. Yang, et al.[7] proposed a DFT architecture that allows testing crypto hardware with high fault coverage yet without exposing hard-coded crypto keys to leakage. Novak and Biasizzo[8] propose a locking mechanism for JTAG with negligible added cost by adding LOCK and KEY registers. To lock the TAP, a key is entered using a special LOCK instruction. To unlock the TAP, a special UNLOCK instruction is issued, which requires shifting in the key via TDI. The TAP state machine is not altered. Compliance with the 1149.1 standard is maintained except for the rejection of standard instructions (including ones that are mandated by 1149.1) while the TAP is locked.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

5 Defenses for JTAG

Our security goals are to ensure the authenticity of the devices in the JTAG chain, to ensure the secrecy of communication between the JTAG master and the chip, and to reject inauthentic JTAG messages. To achieve these goals, we make use of three standard security primitives: a keyed hash function, a stream cipher, and a message authentication code. Using these primitives, we construct protocols that significantly enhance the security of JTAG.

JTAG communication takes place between a master and a chip. The master is typically a microcontroller or a test station. The threat model of the link is similar to the threat model assumed by designers of network security protocols. Since the threats are similar, it is tempting to apply existing solutions for network communication, such as SSH and SSL. Circuit complexity and key distribution are just two of the problems. SSH and SSL require significant amounts of computation for performing public key cryptography. Even on the full-size computers for which the protocols were designed, the crypto arithmetic is performed by software routines, not native opcodes. These big-integer calculations do not lend themselves to lightweight hardware implementation. Furthermore, the semantics of JTAG include prompt synchronous response, so multi-cycle crypto operations would be problematic.

A mechanism that is suitable for securing JTAG is the authentication and key establishment scheme described in [9]. This lightweight scheme uses a PUF [10][11][9][12]. to authenticate the chip and to establish a cipher key for secure communication. The requirements for challenge-response device authentication are repeatability and unpredictability. Given the same challenge, the chip should always calculate the same response. Without having observed the chip's response to a given challenge, it should be impossible for an attacker to predict the response. We make use of a similar idea to the PUF authentication and key establishment scheme, but where fuses and a keyed hash are used instead of a PUF. In a PUF, each chip's uniqueness comes from subtle physical differences between chips. For our keyed hash, the uniqueness comes from fuse bits that are programmed at the factory. Either a PUF or a keyed hash can support a challenge-response protocol for device authentication. There are pros and cons to each of these two mechanisms. Briefly, PUFs have the advantage of being intrinsically unique, not needing uniqueness to be programmed into them. Fuses have the advantage of using less area, and being a a mature technology that is extremely reliable across temperature, aging, and power variations.

For verifying the authenticity of a part in the JTAG chain, a challenge is sent to the chip, which feeds the challenge to its keyed hash function. The response is returned to the tester, where it is compared with the expected response. The challenge is shifted into the chip via the TDI line. The response is shifted out via TDO. A custom JTAG instruction is invoked to trigger the calculation of the hash.

We also require a cipher to encrypt the communication. A block cipher is not suitable due to its large die area. Block ciphers are also problematic in this application because they operate on whole blocks of data, and the delay this introduces conflicts with standard JTAG timing. The preferred mechanism for encrypting JTAG communication is a stream cipher. Robshaw[13] discusses stream ciphers in depth. Our current implementation uses the Trivium stream cipher[14]. Other stream ciphers can be used.

To protect against inauthentic JTAG messages, we need a message authentication code (MAC) scheme. A MAC algorithm uses a key that is known to the sender and receiver to verify that the message was sent by the authentic sender (not spoofed) and was not modified in transit. The most simplistic MAC scheme is just to calculate the hash of the message and the key, concatenated together. This simplistic scheme is not cryptographically strong. Current popular MAC schemes such as HMAC[15] use nested hash functions, each operating on a block of text. Recently, Arazi[16]

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

level	authenticity	secrecy	integrity
0	no	no	no
1	yes	no	no
2	yes	yes	no
3	yes	yes	yes

Figure 10: We define four levels of assurance. Levels correspond to the set of assurances that are provided.

describes the construction of MAC functions from stream ciphers in embedded platforms where computational power is a constraint.

The timing of JTAG makes it highly desirable that the MAC verification scheme produce the answer immediately upon receipt of the final bit of the message. Introducing a delay while a message is authenticated would complicate the timing of the application of test vectors. Therefore, block-based algorithms are not desirable for our purpose. We make use of an incremental MAC function. Incremental hashing can be performed so that computation is being done while the bits of the message are being serially received [17].

Another mechanism that is required for securing JTAG is access control. By access control, normally what is meant is that there is a set of subjects, a set of objects, and at each subject-object intersection, there is a set of permissions. Some basic access control schemes have been proposed for JTAG and some have been included in commodity parts. A rudimentary but strong access control mechanism is a side-effect of using a MAC on the messages. If only the tester has the challenge-response pairs for the keyed hash, only the legitimate tester will be able to negotiate a MAC key, so the MAC algorithm assures that an unauthorized party will not be able to successfully issue test instructions.

This next subsections discuss how the security primitives are used to secure JTAG communication.

5.1 Secure JTAG Communication Protocol

Due to the wide range of cost and security sensitivity for different chips, a single rigid protocol will either fail to provide all of the necessary assurances or, alternatively, add excessive overhead to the cost of the parts. Therefore we define four levels of protection and provide solutions for each of the levels. Chips of varying levels can be freely mixed on the same JTAG chain without interoperability problems.

- **Level 0:** No assurances. This is equivalent to the vast majority of JTAG-enabled chips currently made.
- **Level 1:** Authenticity is assured. The authentic chip is guaranteed to be involved in the JTAG authenticity probe. No further assurances are available.
- **Level 2:** In addition to the assurance provided by Level 1, secrecy of JTAG signals is assured.
- **Level 3:** In addition to the assurance provided by Level 2, the JTAG link is protected against active attacks: insertion, modification, reordering, replay, and deletion of messages.

The JTAG assurance level of each of the chips in the JTAG chain is known by the tester. This attribute can be defined in the BSDL file for the chip. Note that level 0 involves no active participation in the proposed protocol. Level 0 can be considered an explicit provision for backward compatibility. Our scheme allows participating and nonparticipating chips to coexist in the JTAG chain with each part functioning at its full assurance level. The presence of low assurance chips in the chain does not affect the assurances provided by higher assurance level chips in the same chain. It is, however, the designer's responsibility to choose the right JTAG assurance level for his chip and its intended application.

This section describes the protocol used for communication between the tester and the chip.

5.2 Level 0 Protocol

Communication with a level-0 device is exactly the same as what is defined in the 1149.1 standard.

5.3 Level 1 Protocol

Communication with a level-1 device is exactly the same as what is defined in the 1149.1 standard except for addition of an authentication operation.

1. Tester randomly extracts a CRP from storage.
2. Tester sends challenge to chip.
3. Chip applies challenge to keyed hash module.
4. Chip sends result to tester.
5. Tester compares chip's response with CRP.
6. Regular JTAG operations commence.

5.4 Level 2 Protocol

Communication with a level-2 device involves sending and receiving encrypted data to and from the data register (DR). The encryption only affects the signal on the TDI and TDO lines. The TCK, TMS, and TRST lines retain the exact semantics defined in the 1149.1 standard. The TAP controller's JTAG state machine is also unaffected. Level-2 communication occurs in two phases: setup and communication.

The purpose of the setup phase is to establish a shared secret between the tester and the chip. The setup phase proceeds as follows:

1. Tester randomly selects and extracts a CRP.
2. Tester sends challenge to chip.
3. Chip applies challenge to keyed hash module.
4. Chip uses response as key for stream cipher.

In the case of the tester writing to the data register on the chip, the protocol is as follows:

1. Tester encrypts JTAG DR contents using shared secret.
2. Tester places chip into SHIFT-DR state.
3. Tester shifts in the encrypted contents of DR.

4. Chip decrypts data into plaintext register as it is shifted in.
5. Tester places chip into EXIT-IR or EXIT-DR state.
6. Plaintext register is latched in and used.

The bits that are shifted out via TDO during a tester-writes-to-chip operation are ciphertext. The protocol for the tester to read data from the chip is as follows:

1. Tester initiates read operation.
2. Chip encrypts contents of DR using stream cipher as it transmits the bits on TDO.
3. Tester decrypts bits using stream cipher.

5.5 Level 3 Protocol

The level-3 protocol has two phases: setup and communication. The setup phase establishes the secrets that are shared between the tester and the chip for the cipher and MAC algorithms. The communication phase encapsulates, transmits, and de-encapsulates the contents of the IR and DR.

The behavior at level 3 is the same as level 2, but an additional key establishment operation is performed for the MAC key. Using that additional MAC key, the chip calculates the MAC of the bits are received. There are two distinct ways of doing this. First is to use an incremental MAC algorithm of the form shown by Lai[17]. The second is to use a more conventional MAC scheme that requires full blocks of data before commencing MAC processing. The MAC is checked when the TAP transitions out of the SHIFT-DR state. If the MAC passes, the contents of DR are transferred to a VALIDATED_DR register and a flag on VALIDATED_DR is set, indicating that the contents of the register has been validated. Later operations can make use of the contents of the VALIDATED_DR register. On one hand, this approach adds area overhead. On the other hand, it enables the use of a conventional (non-incremental) MAC algorithm, the kind that has been more extensively studied.

The steps of the MAC setup phase are the same as the crypto key setup in the level-2 protocol. The keyed hash is used along with challenge-response pairs that are stored by the tester. Rudimentary access control is provided by virtue of MACs being checked. Only the authentic sender can successfully negotiate a MAC key.

6 Costs and Benefits Associated with JTAG Defenses

Successful mechanisms for enhancing the security of JTAG must satisfy cost and operational constraints, in addition to providing the required forms of assurances.

6.1 Die Area Overhead

Some of the systems we are protecting are high-volume commodity items which are price-sensitive. We cannot add more than 10% to the die area of any protected part. We see in figure 11 that for designs of non-trivial die area, the proposed schemes will not add significant cost. From figure 11 we see that for a design that uses 10,000 slices, the area overhead is less than 9% even for level 3 defenses, and about 4% for level 1. 10,000 slices of a Xilinx Spartan 3e FPGA is equivalent to 1.4 million gates [18].

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

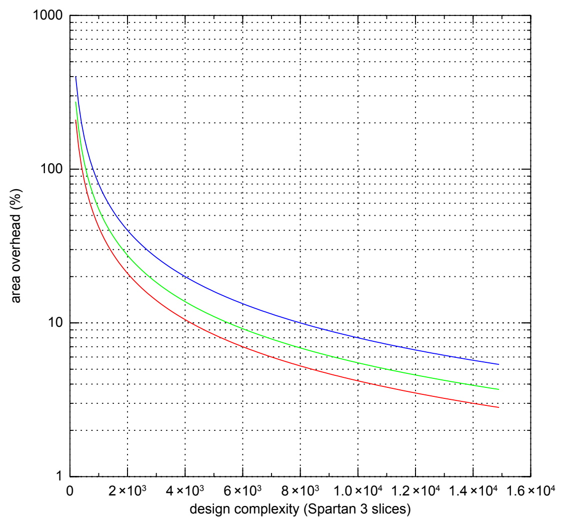

Figure 11: Area overhead is shown for the protection levels 1 through 3, from bottom to top. The cost of the security enhancements independent of design complexity, so the percentage overhead is lower for more complex designs. The four levels of assurance provide progressively higher levels of assurance. An indication of the area cost of each protection level is given by the number of additional FPGA slices used by the enhanced JTAG circuitry. These figures are for a Spartan 3e. There are no fuses in the FPGA so fuses are modeled as hard-coded bit vectors. Overhead in an ASIC will be less.

6.2 Test Time Overhead

There is test time overhead associated with the security enhancements we propose. For the level 1 defenses, the time overhead is as follows:

1. Time to shift in challenge. (80 cycles of test clock)
2. Time to initialize stream cipher. (1152 cycles of functional clock) [2]
3. Time to shift out response. (80 cycles of test clock)

For the level 2 defenses, the time overhead is spent once per test session, to set up the cipher key and initialize the stream cipher.

1. Shift in the challenge (80 cycles of test clock).
2. Initialize stream cipher 1 using challenge (1152 cycles of functional clock).
3. Extract 80 bits of keystream from cipher 1 (80 cycles of functional clock).
4. Initialize stream cipher 2 using 80 bits from cipher 1 (1152 cycles of functional clock).

After this setup, there is no test time overhead associated with the level-2 defenses. Stream cipher 2 operates synchronously with the test clock to encrypt and decrypt data. For level 3 defenses, the minimum additional test time overhead is the same as for level 2.

[2]The stream cipher we use in our reference implementation is Trivium[14]; it uses an 80-bit key and takes 1152 cycles to initialize.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

6.3 Operational Costs

The security enhancements that we propose require that challenge-response pairs be extracted at manufacturing test time, before the chips leave the manufacturing facility. If fuses are used, as opposed to a PUF, the fuses should be blown with a random pattern prior to the extraction of challenge-response pairs. For each chip that the customer receives, he needs the challenge-response pairs. These have to be delivered to the customer over a secure channel.

It is also possible to eliminate the burden of keeping track of the identity of each chip on a reel of parts. Instead of querying the chip maker for the challenge-response pairs for a part, the chip can contain a unique code which can then be used for retrieving the challenge response pairs from the chip maker.

6.4 Impact of Defenses on Known Threats

The proposed defenses protect against a range of attacks discussed in this paper. We will now illustrate how two otherwise practical attacks can be prevented.

- Attacker Obtains Embedded Secrets: The attacker could either sniff the JTAG link as the key is being programmed, or the attacker could perform an unauthorized debugging session to read out the key. These attacks are popular in hacking satellite TV boxes to extract the box keys using JTAG [6] and in taking a snapshot of the firmware in the box [19]. Novak and Biasizzo[8] propose a solution to unauthorized JTAG access based on access control. A key must be presented to the chip before it can be manipulated with JTAG. Under an expansive threat model, where hostile chips may be present in the JTAG chain, the scheme is not effective. On the other hand, the proposed level-2 and level-3 defenses are comprehensive and guard against passive sniffing by encrypting the link. They also prevent unauthorized debugging as a side effect of MACing the data.

- Cloning an existing system would involve different attacks depending on the implementation style of the system. If it is a CPU- or PLD-based system, cloning would involve reading out firmware or a bitstream. The attacks are similar to obtaining an embedded secret , and the defenses are the same. For example, at levels 2 and 3, the attacker will need to negotiate keys with the chip in order to communicate over JTAG, and this will be impossible for the attacker since he lacks the challenge-response pairs that were set up offline.

7 Conclusion

In systems where sensitive data are transported or accessible via JTAG, we do not recommend using a daisy-chain topology with conventional unprotected JTAG. A star topology protects against many of the attacks discussed in this article. However, the star topology increases PCB complexity and cost and doesn't eliminate all JTAG-based system vulnerabilities. An alternative is to retain the daisy-chain topology, and to derive needed security from cryptographic enhancements built on top of JTAG.

We have devised a scheme that provides a significant improvement in JTAG security with reasonable added cost. The scheme is flexible in the sense that it can provide high assurance for important chips and lower assurance (and lower cost) for less important chips. Compatibility is maintained across the assurance levels and compatibility is maintained with the IEEE 1149.1 JTAG standard to the maximum extent possible.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

The scope of this work is restricted to JTAG. There is a multitude of other threats to hardware that do not involve JTAG. For instance, exploits via bus probing have been successfully executed in the past[20] and will probably be used again. However, it should be noted that the original reason for the existence of JTAG was to facilitate boundary scan of I/O pins because it was difficult to probe the wiring on modern printed circuit boards. Securing JTAG denies the attacker what would otherwise be an easy way of probing a bus. We do not address several other threats including the threat of manufacturers inserting hostile functionality in the chips that they supply. Detecting the presence of such functionality is an active research topic [21][22].

References

[1] IEEE Std 1149.1-2001, Test Access Port and Boundary-Scan Architecture.

[2] IEEE Std 1532-2002, In-System Configuration of Programmable Devices.

[3] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *Proceedings of the IEEE Int Test Conference*, 2004, pp. 339–344.

[4] The Dishnewbies Team. Jtag guide. [Online]. Available: http://dishnewbies.com

[5] satcardsrus.com. Secure loading advanced blocker 3m's for rom images. [Online]. Available: http://www.satcardsrus.com/dish_net%203m.htm

[6] The Dishnewbies Team. JTAG guide. [Online]. Available: http://dishnewbies.com/jtag.shtml

[7] B. Yang, R. Karri, and K. Wu, "Secure scan: a design-for-test architecture for crypto chips," in *Proceedings of the IEEE/ACM Design Automation Conference*, 2005, pp. 135–140.

[8] F. Novak and A. Biasizzo, "Security extension for IEEE Std 1149.1," *Journal of Electronic Testing*, vol. 22, no. 3, pp. 301–303, 2006.

[9] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of IEEE/ACM Design Automation Conference*, June 2007, pp. 9–14.

[10] R. Pappu, "Physical one-way functions," Massachusetts Institute of Technology, Tech. Rep., 2001.

[11] B. Gassend, D. Clarke, M. V. Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the ACM Computer and Communication Security Conference*, 2002, pp. 148–160.

[12] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proceedings of IEEE International Test Conference*, Oct. 2008, pp. 1–10.

[13] M. J. B. Robshaw, "Stream ciphers," RSA Laboratories, Tech. Rep. TR-701, 1995.

[14] C. D. Canniere and B. Preneel, "Trivium specifications," *ECRYPT Stream Cipher Project*, 2006.

[15] R. Canetti, "HMAC: keyed-hashing for message authentication," *RFC*, vol. 2104, p. 2104, 1997.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE

[16] B. Arazi, "Message authentication in computationally constrained environments," *IEEE Transactions on Mobile Computing*, vol. 8, no. 7, pp. 968–974, July 2009.

[17] X. Lai, R. Rueppel, and J. Woollven, "A fast cryptographic check-sum algorithm based on stream ciphers," in *Advances in Cryptology-AusCrypt*. Springer-Verlag, 1992, pp. 339–348.

[18] Xilinx. Spartan-3e fpga family: Introduction and ordering information. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[19] Fwaggle. Howto: JTAG interface on a Dish 3700 receiver. [Online]. Available: http://www.hungryhacker.com/articles/misc/dish3700_jtag

[20] A. Huang, "Keeping secrets in hardware: The Microsoft XBox case study," in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, 2002, pp. 213–227.

[21] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust*, June 2008.

[22] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proceedings of IEEE Symposium on Security and Privacy*, May 2007, pp. 296–310.

Digital Object Indentifier 10.1109/MDT.2009.161 0740-7475/$26.00 © 2009 IEEE